

```
/* BLIST Module for LISTEN Program */
/* Disk Functions */
/* Copyright 1991-9 by James Hoyt Clark */
```

```
#include <stdio.h>
#include <stdlib.h>
#include <graphics.h>
#include <io.h>
#include <fcntl.h>
#include "twindow.h"
#include "keys.h"
```

```
extern struct holdrecord
{
    char name[31];
    char id[8];
    unsigned char num0;
    unsigned char num1;
    unsigned char num2;
    unsigned char num3;
    unsigned char num4;
    unsigned char num5;
    unsigned char num6;
    unsigned char num7;
    unsigned char num8;
    unsigned char num9;
    unsigned char potency[11];
    unsigned char potnum[3];
    unsigned char max;
    unsigned char min;
    unsigned char rise;
    unsigned char fall;
    unsigned char tag;
};
```

```
extern struct holdrecord hold[90];
```

```
extern struct filrecord1
{
    char name[31];
    char id[8];
    unsigned char num0;
    unsigned char num1;
    unsigned char num2;
    unsigned char num3;
    unsigned char num4;
```

```

        unsigned char num5;
        unsigned char num6;
        unsigned char num7;
        unsigned char num8;
        unsigned char num9;
        unsigned char potency[11];
        unsigned char potnum[3];
        unsigned char tag;
    };
extern struct filrecord1 filter1[50];

extern struct filrecord2
{
    char name[31];
    char id[8];
    unsigned char num0;
    unsigned char num1;
    unsigned char num2;
    unsigned char num3;
    unsigned char num4;
    unsigned char num5;
    unsigned char num6;
    unsigned char num7;
    unsigned char num8;
    unsigned char num9;
    unsigned char potency[11];
    unsigned char potnum[3];
    unsigned char tag;
};
extern struct filrecord2 filter2[50];

extern struct potrecord
{
    char name[8];
    char num[6];
    char CR;
    char LF;
};
extern struct potrecord pot[1000];

extern void (*helpfunc)();
extern void help();
extern char dilutionflag;
extern char outname[31];
extern char outid[8];

```

```
extern unsigned char outnum0;
extern unsigned char outnum1;
extern unsigned char outnum2;
extern unsigned char outnum3;
extern unsigned char outnum4;
extern unsigned char outnum5;
extern unsigned char outnum6;
extern unsigned char outnum7;
extern unsigned char outnum8;
extern unsigned char outnum9;
extern unsigned char outdilution;
extern int potpoint;
extern int npot;
extern int dilindex;
extern char ifgraphic;
extern char filteroutflag;
extern char whichallergy;
extern char h;
extern char fcnt1;
extern char fcnt2;
extern int nhold;
extern int nfilter;
extern char pnum;
extern char ntags;
extern char nfltags;
extern char nf2tags;
extern int reportpoint;
extern long int npoint;
extern char pointflag;
extern int pointcolor;
extern char outholdflag;
extern int basecolor;
extern int letter1color;
extern int letter2color;
extern int letter3color;
extern int accentcolor;
extern char whichDRIVE;
extern int usedeletes;
extern int defaultvalues[100];
extern unsigned char max;
extern unsigned char min;
extern unsigned char rise;
extern unsigned char fall;
extern char fromkey;
extern long int start;
```

```
extern char msg_line[80];
extern char board;
extern int wavepoint;
extern unsigned char cPotNum[3];
```

```
char name[31];
char id[8];
unsigned char num0;
unsigned char num1;
unsigned char num2;
unsigned char num3;
unsigned char num4;
unsigned char num5;
unsigned char num6;
unsigned char num7;
unsigned char num8;
unsigned char num9;
unsigned char dilution;
char lastdilution;
char biblio;
int num[5];
```

```
static int dil0[13] = {0,5,6,7,8,9,10,13,15,16,17,99,10};
static int dil1[13] = {0,6,7,8,9,10,13,15,16,17,18,99,10};
static int dil2[13] = {0,8,9,13,15,16,17,18,19,20,21,99,10};
static int dil3[13] = {0,10,11,13,15,16,17,18,19,20,21,99,10};
static int dilz[13] = {0,3,4,5,6,7,8,9,10,13,15,99,10};
static int dilr[18] = {0,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,99,15};
static int dils[20] = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,23,24,25,26,99,17};
static int caps[9] = {0,55,56,57,58,59,60,99,6};
static int drop[23] = {0,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,
77,78,79,80,99,20};
/*static int dilq[134] = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,
21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,
39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,
57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,
75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,
93,94,95,96,97,98,99,100,101,102,103,104,105,106,107,108,
109,110,111,112,113,114,115,116,117,118,119,120,121,
122,123,124,125,126,127,128,129,130,131,999,131};*/
static int dilq[289] = {0,1,2,3,4,5,6,7,8,9,10,
11,12,13,14,15,16,17,18,19,20,
21,22,23,24,25,26,27,28,29,30,
31,32,33,34,35,36,37,38,39,40,
```



```

41,42,43,44,45,46,47,48,49,50,
51,52,53,54,55,56,57,58,59,60,
61,62,63,64,65,66,67,68,69,70,
71,72,73,74,75,76,77,78,79,80,
81,82,83,84,85,86,87,88,89,90,
91,92,93,94,95,96,97,98,99,100,
101,102,103,104,105,106,107,108,109,110,
111,112,113,114,115,116,117,118,119,120,
121,122,123,124,125,126,127,128,129,130,
131,132,133,134,135,136,137,138,139,140,
141,142,143,144,145,146,147,148,149,150,
151,152,153,154,155,156,157,158,159,160,
161,162,163,164,165,166,167,168,169,170,
171,172,173,174,175,176,177,178,179,180,
181,182,183,184,185,186,187,188,189,190,
191,192,193,194,195,196,197,198,199,200,
201,202,203,204,205,206,207,208,209,210,
211,212,213,214,215,216,217,218,219,220,
221,222,223,224,225,226,227,228,229,230,
231,232,233,234,235,236,237,238,239,240,
241,242,243,244,245,246,247,248,249,250,
251,252,253,254,255,256,257,258,259,260,
261,262,263,264,265,266,267,268,269,270,
271,272,273,274,275,276,277,278,279,280,
281,282,283,284,285,286,
999,286};

```

```

void getproduct(long int position)
{
char diskbuffer[51];
int fhandle,i,j;
if ((fhandle=open("MASTER.DAT",O_RDONLY|O_BINARY)) == -1) {
    printf("open failed");
    exit(1);
}
lseek(fhandle,51*position,SEEK_CUR);
if ((read(fhandle,diskbuffer,51)) == -1) {
    perror("read error");
    exit(1);
}
strncpy(name,diskbuffer,31);
j=0;
for (i=31; i<39; i++) {
    id[j]=diskbuffer[i];
    j++;
}

```

```

    }
    num0 = diskbuffer[39];
    num1 = diskbuffer[40];
    num2 = diskbuffer[41];
    num3 = diskbuffer[42];
    num4 = diskbuffer[43];
    num5 = diskbuffer[44];
    num6 = diskbuffer[45];
    num7 = diskbuffer[46];
    num8 = diskbuffer[47];
    num9 = diskbuffer[48];
    dilution = diskbuffer[49];
    biblio = diskbuffer[50];
    close(fhandle);
}

```

/* Disk Operations */

```

void readholdfile()
{
    FILE *fptr;
    h=0;
    if( (fptr=fopen("C:\\WORK\\HOLD.DAT","rb"))==NULL ) {
        operationinfo("errorfl",10,5,yeskey);
        exit(0);
    }
    else {
        while(fread(&hold[h],sizeof(hold[0]),1,fptr)==1 ) h++;
        fclose(fptr);
    }
    h--;
}

```

```

void writeholdfile()
{
    FILE *fptr;
    if ((fptr=fopen("C:\\WORK\\HOLD.DAT","wb"))==NULL ) {
        operationinfo("errorfl",10,5,yeskey);
        exit(0);
    }
    else {
        fwrite(hold,sizeof(hold[0]),h+1,fptr);
        fclose(fptr);
    }
}

```

```

void readfilter1()
{
FILE *fptr;
fcnt1=0;
if ((fptr=fopen("C:\\WORK\\FILTER1.DAT","rb"))==NULL ) {
    operationinfo("errorfl",10,5,yeskey);
    exit(0);
}
else {
    while(fread(&filter1[fcnt1],sizeof(filter1[0]),1,fptr)==1 ) fcnt1++;
    fclose(fptr);
}
fcnt1--;
}

```

```

void writefilter1()
{
FILE *fptr;
int flcount=fcnt1+1;
if ((fptr=fopen("C:\\WORK\\FILTER1.DAT","wb"))==NULL ) {
    operationinfo("errorfl",10,5,yeskey);
    exit(0);
}
else {
    fwrite(filter1,sizeof(filter1[0]),flcount,fptr);
    fclose(fptr);
}
}

```

```

void readfilter2()
{
FILE *fptr;
fcnt2=0;
if ((fptr=fopen("C:\\WORK\\FILTER2.DAT","rb"))==NULL ) {
    operationinfo("errorfl",10,5,yeskey);
    exit(0);
}
else {
    while(fread(&filter2[fcnt2],sizeof(filter2[0]),1,fptr)==1 ) fcnt2++;
    fclose(fptr);
}
fcnt2--;
}

```

```

void writefilter2()

```

```

{
FILE *fptr;
int f2count=fcnt2+1;
if((fptr=fopen("C:\\WORK\\FILTER2.DAT","wb"))==NULL ) {
    operationinfo("errorfl",10,5,yeskey);
    exit(0);
}
else {
    fwrite(filter2,sizeof(filter2[0]),f2count,fptr);
    fclose(fptr);
}
}

```

```

void clear_all()
{
long int j;
int key;
key=operationinfo("clrvst ",4,4,yeskey);
if (key == 'Y') {
    usedeletes++;
    defaultvalues[28]=usedeletes;
    writedefaults();
    for (j=0; j<npoint; j++) zeropoint(j);
    writepointfile();
    h=-1;
    reportpoint=-1;
    pointflag=0;
    outholdflag=0;
    pointcolor=basecolor;
    display_points();
    setpointwindow();
}
else tellcanceled();
}

```

/* Dilution */

```

void getdilution()
{
if (dilutionflag) {
    if (potpoint < 0) potpoint=286; //???????131
    if (potpoint > dilq[286]) potpoint=0; //131
    dilindex=dilq[potpoint];
    potencynum();
    return;
}
}

```

```

    }
    if (outdilution == '0') {
        if (potpoint < 0) potpoint=10;
        if (potpoint > dil0[12]) potpoint=0;
        dilindex=dil0[potpoint];
    }
    if (outdilution == '1') {
        if (potpoint < 0) potpoint=10;
        if (potpoint > dil1[12]) potpoint=0;
        dilindex=dil1[potpoint];
    }
    if (outdilution == '2') {
        if (potpoint < 0) potpoint=10;
        if (potpoint > dil2[12]) potpoint=0;
        dilindex=dil2[potpoint];
    }
    if (outdilution == '3') {
        if (potpoint < 0) potpoint=10;
        if (potpoint > dil3[12]) potpoint=0;
        dilindex=dil3[potpoint];
    }
    if (outdilution == 'Z') {
        if (potpoint < 0) potpoint=10;
        if (potpoint > dilz[12]) potpoint=0;
        dilindex=dilz[potpoint];
    }
    if (outdilution == 'A') {
        if (whichallergy) {
            if (potpoint < 0) potpoint=15;
            if (potpoint > dilr[17]) potpoint=0;
            dilindex=dilr[potpoint];
        }
        else outdilution='S';
    }
    if (outdilution == 'S') {
        if (potpoint < 0) potpoint=17;
        if (potpoint > dils[19]) potpoint=0;
        dilindex=dils[potpoint];
    }
    if (outdilution == 'c') {
        if (potpoint < 0) potpoint=6;
        if (potpoint > caps[8]) potpoint=0;
        dilindex=caps[potpoint];
    }
    if (outdilution == 'd') {

```

```

        if (potpoint < 0) potpoint=20;
        if (potpoint > drop[22]) potpoint=0;
        dilindex=drop[potpoint];
    }
    if (outdilution == '/') {
        potpoint=0;
        dilindex=dilr[potpoint];
    }
    potencynum();
}

void displaydilution(WINDOW *wnd_dilution,int i)
{
    char sName[8];
    int j;
    for (j=0; j<8; j++)
        sName[j] = pot[dilindex].name[j];
    sName[7] = 0;
    wprintf(wnd_dilution," %s",sName);
    if (i % 8 == 0) wprintf(wnd_dilution,"\n");
}

void switchdilution(int XCM)
{
    int i;
    for (i=0; i<npot; i++) {
        if (pot[i].name[1] == XCM) {
            potpoint=i;
            break;
        }
    }
    putchar(BELL);
    getpot(potpoint);
}

void alldilutions()
{
    WINDOW *wnd_dilution;
    int i,key;
    setmenuhelp("dilutn ",10,10);
    wnd_dilution = establish_window(3,3,22,77);
    set_colors(wnd_dilution,ALL,GREEN,BLACK,DIM);
    set_colors(wnd_dilution,ACCENT,WHITE,BLACK,DIM);
    display_window(wnd_dilution);
    i=0;

```

```

dilindex=0;
if (dilutionflag) {
    showline("msg_MasterDiluti");
    wprintf(wnd_dilution,msg_line);
    while (dilindex != 999) {
        dilindex=dilq[i];
        if (dilindex != 999) displaydilution(wnd_dilution,i);
        i++;
    }
    wprintf(wnd_dilution,"\n");
}

i=0;
dilindex=0;
if (outdilution == '0') {
    showline("msg_ArnicaKUFSta");
    wprintf(wnd_dilution,msg_line);
    while (dilindex != 99) {
        dilindex=dil0[i];
        if (dilindex != 99) displaydilution(wnd_dilution,i);
        i++;
    }
}

if (outdilution == '1') {
    showline("msg_ArnicaKUF1 ");
    wprintf(wnd_dilution,msg_line);
    while (dilindex != 99) {
        dilindex=dil1[i];
        if (dilindex != 99) displaydilution(wnd_dilution,i);
        i++;
    }
}

if (outdilution == '2') {
    showline("msg_ArnicaKUF2 ");
    wprintf(wnd_dilution,msg_line);
    while (dilindex != 99) {
        dilindex=dil2[i];
        if (dilindex != 99) displaydilution(wnd_dilution,i);
        i++;
    }
}

if (outdilution == '3') {
    showline("msg_ArnicaKUF3 ");
    wprintf(wnd_dilution,msg_line);
    while (dilindex != 99) {
        dilindex=dil3[i];

```

```

        if (dilindex != 99) displaydilution(wnd_dilution,i);
        i++;
    }
}

if (outdilution == 'Z') {
    showline("msg_ArnicaZ  ");
    wprintf(wnd_dilution,msg_line);
    while (dilindex != 99) {
        dilindex=dilz[i];
        if (dilindex != 99) displaydilution(wnd_dilution,i);
        i++;
    }
}

if (outdilution == 'A') {
    if (whichallergy) {
        showline("msg_RinkelAllerg");
        wprintf(wnd_dilution,msg_line);
        while (dilindex != 99) {
            dilindex=dilr[i];
            if (dilindex != 99) displaydilution(wnd_dilution,i);
            i++;
        }
    }
    else outdilution='S';
}

if (outdilution == 'S') {
    showline("msg_StandardLow ");
    wprintf(wnd_dilution,msg_line);
    while (dilindex != 99) {
        dilindex=dils[i];
        if (dilindex != 99) displaydilution(wnd_dilution,i);
        i++;
    }
}

if (outdilution == 'c') {
    showline("msg_Capsule  ");
    wprintf(wnd_dilution,msg_line);
    while (dilindex != 99) {
        dilindex=caps[i];
        if (dilindex != 99) displaydilution(wnd_dilution,i);
        i++;
    }
}

if (outdilution == 'd') {
    showline("msg_Drop      ");

```



```

wprintf(wnd_dilution,msg_line);
while (dilindex != 99) {
    dilindex=drop[i];
    if (dilindex != 99) displaydilution(wnd_dilution,i);
    i++;
}
}
if (outdilution == '/') {
    showline("msg_CombinationO");
    wprintf(wnd_dilution,msg_line);
    dilindex=0;
    displaydilution(wnd_dilution,0);
}
key=0;
while (key != ESC) {
    key = get_char();
    key = toupper(key);
    switch(key) {
        case ',': prevpot(); break;
        case '.': nxtpot(); break;
        case 'Z': fromkey='3';
                    potpoint=scan(1);
                    helpfunc=help;
                    getpot(potpoint);
                    fromkey='I';
                    break;
        case 'A': getpot(0);
                    break;
        case INS: setpot();
                    break;
        case UP:
                    break;
        case DN:
                    break;
        default: break;
    }
}
delete_window(wnd_dilution);
}

/* Set for Output */

void loadout()
{
    strcpy(outname,name);

```

```

strcpy(outid,id);
outnum0=num0;
outnum1=num1;
outnum2=num2;
outnum3=num3;
outnum4=num4;
outnum5=num5;
outnum6=num6;
outnum7=num7;
outnum8=num8;
outnum9=num9;
outdilution=dilution;
if (lastdilution != dilution) {
    if (ifgraphic) displaypotgraph();
    else getpot(potpoint);
    lastdilution=dilution;
}
}

/* Hold */

void counttags()
{
int i;
ntags=0;
for (i=0; i<h+1; i++) if (hold[i].tag) ntags++;
}

void holduntags()
{
int i;
for (i=0; i<nhold+1; i++) hold[i].tag = 0;
}

void allholdtags()
{
int i;
for (i=0; i<nhold+1; i++) hold[i].tag = 1;
}

void deletehold()
{
int c;
if (h < 0) {
    twobells();
}
}

```

```

        return;
    }
    c=operationinfo("clrHold",10,10,yeskey);
    if (c == 'Y') {
        h = -1;
        operationinfo("Holdclr",10,10,nokey);
    }
    else tellcanceled();
}

void deleteonehold(int hpt)
{
    int ptr;
    for (ptr=hpt; ptr<h+1; ptr++) {
        strcpy(hold[ptr].name,hold[ptr+1].name);
        strcpy(hold[ptr].id,hold[ptr+1].id);
        strcpy(hold[ptr].potency,hold[ptr+1].potency);
        hold[ptr].potnum[0] = hold[ptr+1].potnum[0];
        hold[ptr].potnum[1] = hold[ptr+1].potnum[1];
        hold[ptr].potnum[2] = hold[ptr+1].potnum[2];
        hold[ptr].num0 = hold[ptr+1].num0;
        hold[ptr].num1 = hold[ptr+1].num1;
        hold[ptr].num2 = hold[ptr+1].num2;
        hold[ptr].num3 = hold[ptr+1].num3;
        hold[ptr].num4 = hold[ptr+1].num4;
        hold[ptr].num5 = hold[ptr+1].num5;
        hold[ptr].num6 = hold[ptr+1].num6;
        hold[ptr].num7 = hold[ptr+1].num7;
        hold[ptr].num8 = hold[ptr+1].num8;
        hold[ptr].num9 = hold[ptr+1].num9;
        hold[ptr].max = hold[ptr+1].max;
        hold[ptr].min = hold[ptr+1].min;
        hold[ptr].rise = hold[ptr+1].rise;
        hold[ptr].fall = hold[ptr+1].fall;
        hold[ptr].tag = hold[ptr+1].tag;
        /*
        strcpy(holdv[ptr].ptname,holdv[ptr+1].ptname);
        holdv[ptr].voltage=holdv[ptr+1].voltage;
        strcpy(holdv[ptr].freqname,holdv[ptr+1].freqname);
        holdv[ptr].wavepoint=holdv[ptr+1].wavepoint;
        holdv[ptr].y=holdv[ptr+1].y;
        holdv[ptr].z=holdv[ptr+1].z;*/
    }
    h--;
}

```

```

void viewhold()
{
int i,j,temp,key=0;
int hpoint,start,end;
WINDOW *wnd_hold;
char lineinfo=1;
setmenuhelp("hold ",0,0);
wnd_hold = establish_window(0,0,25,80);
showline("msg_HoldList ");
set_title(wnd_hold,msg_line);
set_colors(wnd_hold,ALL,letter3color,AQUA,DIM);
set_colors(wnd_hold,ACCENT,accentcolor,WHITE,BRIGHT);
display_window(wnd_hold);
hpoint=0;
start=0;
while (key != ESC) {
    wcursor(wnd_hold,0,0);
    if (start+20 <= h) end = start+20;
    else end = h+1;
    for (j=start; j<end; j++) {
        if (j == hpoint) reverse_video(wnd_hold);
        if (hold[j].tag == 1) temp=16;
        else temp = ' ';
        wputchar(wnd_hold,temp);
        wprintf(wnd_hold,"%#2d. %s %s",j+1,hold[j].name,hold[j].id);
        if (lineinfo) {
            wprintf(wnd_hold," %#3d",hold[j].max);
            wprintf(wnd_hold," %#3d",hold[j].min);
            wprintf(wnd_hold," %#3d",hold[j].rise);
            wprintf(wnd_hold," %#3d",hold[j].fall);
            wprintf(wnd_hold," %s",hold[j].potency);
        }
        if (end != j) wprintf(wnd_hold,"\n");
        normal_video(wnd_hold);
    }
    counttags();
    wcursor(wnd_hold,10,22);
    showline("msg_HoldTagLimit");
    wprintf(wnd_hold,msg_line,h+1,ntags,nhold+1);
    key = get_char();
    key = toupper(key);
    switch(key) {
        case ' ': if (hold[hpoint].tag == 1) hold[hpoint].tag=0;
                  else hold[hpoint].tag = 1;
        case RSWITCH:

```

```

case DN: hpoint++;
        if (hpoint > h) {
            hpoint=0;
            start=0;
        }
        if (hpoint > start+19) {
            start = hpoint;
            clear_window(wnd_hold);
        }
        break;
case LSWITCH:
case UP: hpoint--;
        if (hpoint < 0) {
            hpoint=h;
            start=h-19;
            if (start < 0) start=0;
        }
        if (hpoint < start) {
            start=hpoint-19;
            clear_window(wnd_hold);
            if (start < 0) {
                start=0;
                hpoint=0;
            }
        }
        break;
case PGUP: start=start-20;
        if (start < 0) start=0;
        clear_window(wnd_hold);
        hpoint=start+19;
        break;
case PGDN: start = start+20;
        if (start > h) start=0;
        clear_window(wnd_hold);
        hpoint=start;
        break;
case HOME: start=0;
        hpoint=0;
        break;
case END: start=h-19;
        if (start < 0) start=0;
        hpoint=h;
        break;
case 'W': display_time(); break;
case 'U': holduntags(); break;

```

```

case 'T': allholdtags(); break;
case 'V': if (lineinfo) lineinfo=0;
          else lineinfo++;
          clear_window(wnd_hold);
          break;
case 'R': for (j=0; j<h+1; j++) {
          if (hold[j].tag == 1) hold[j].tag = 0;
          else hold[j].tag = 1;
          }
          break;
case '-': deletehold();
          clear_window(wnd_hold);
          break;
case DEL: if (h < 0) break;
          if (hold < 0) break;
          deleteonehold(hpoint);
          beep();
          clear_window(wnd_hold);
          if (hpoint > h) {
              hpoint--;
              start = hpoint-19;
              if (start<0) start=0;
          }
          writeholdfile();
          writeholdvfile();*/
          break;
case ALT_U: if (h < 0) break;
            if (hold < 0) break;
            for (i=0; i<h+1; i++) {
                if (hold[i].tag == 0) {
                    deleteonehold(i);
                    i--;
                }
            }
            beep();
            clear_window(wnd_hold);
            hpoint=0;
            start=0;
            writeholdfile();
            writeholdvfile();*/
            break;
case 'P': pnum=0;
          printing(); break;
case 'C': OutputCapsule();
          break;

```

```

        case ALT_P: pnum=0;
                    printtag(); break;
        default: break;
    }
}
delete_window(wnd_hold);
}

/* Filter Operations */

void countfltags()
{
    int i;
    nfltags=0;
    for (i=0; i<fcnt1+1; i++) if (filter1[i].tag) nfltags++;
}

void fluntags()
{
    int i;
    for (i=0; i<nfilter+1; i++) filter1[i].tag = 0;
}

void allfltags()
{
    int i;
    for (i=0; i<nfilter+1; i++) filter1[i].tag = 1;
}

void viewfilter1()
{
    int j,temp,key=0;
    long int fpoint,fstart,fend;
    int prtstatus;
    char string[50];
    WINDOW *wnd_filter;

    setmenuhelp("filter ",0,0);
    wnd_filter = establish_window(39,0,25,41);
    showline("msg_Filter1Sel ");
    set_title(wnd_filter,msg_line);
    set_colors(wnd_filter,ALL,BROWN,BLACK,DIM);
    set_colors(wnd_filter,ACCENT,accentcolor,BROWN,BRIGHT);
    display_window(wnd_filter);
    fpoint=0;

```

```

fstart=0;
filteroutflag=1;
fromkey='1';
while (key != ESC) {
    wcursor(wnd_filter,0,0);
    if (fstart+20 <= fcnt1) fend = fstart+20;
    else fend = fcnt1+1;
    for (j=fstart; j<fend; j++) {
        if (j == fpoint) reverse_video(wnd_filter);
        if (filter1[j].tag == 1) temp=16;
        else temp = ' ';
        wputchar(wnd_filter,temp);
        wprintf(wnd_filter,"%#2d.%%s%%s",
            j+1,filter1[j].name,filter1[j].potency);
        if (fend != j) wprintf(wnd_filter,"\n");
        normal_video(wnd_filter);
    }
    countfltags();
    wcursor(wnd_filter,0,22);
    showline("msg_FiltersTL  ");
    wprintf(wnd_filter,msg_line,fcnt1+1,nfltags,nfilter+1);
    key = get_char();
    if (key == 0) key=getch();
    else key = toupper(key);
    switch(key) {
        case ' ': if (filter1[fpoint].tag == 1) filter1[fpoint].tag=0;
                  else filter1[fpoint].tag = 1;
        case RSWITCH:
        case DN: fpoint++;
                 if (fpoint > fcnt1) {
                     fpoint=0;
                     fstart=0;
                 }
                 if (fpoint > fstart+19) {
                     fstart = fpoint;
                     clear_window(wnd_filter);
                 }
                 break;
        case LSWITCH:
        case UP: fpoint--;
                 if (fpoint < 0) {
                     fpoint=fcnt1;
                     fstart=fcnt1-19;
                     if (fstart < 0) fstart=0;
                 }
    }
}

```



```

        if (fpoint < fstart) {
            fstart=fpoint-19;
            clear_window(wnd_filter);
            if (fstart < 0) {
                fstart=0;
                fpoint=0;
            }
        }
        break;
case PGUP: fstart=fstart-20;
        if (fstart < 0) fstart=0;
        clear_window(wnd_filter);
        fpoint=fstart;
        break;
case PGDN: fstart = fstart+20;
        if (fstart > fcnt1) fstart=0;
        clear_window(wnd_filter);
        fpoint=fstart;
        break;
case HOME: fstart=0;
        fpoint=0;
        break;
case END: fstart=fcnt1-19;
        if (fstart < 0) fstart=0;
        fpoint=fcnt1;
        break;
case 'W': display_time(); break;
case 'U': fluntags(); break;
case 'T': allfltags(); break;
case '+': downitem(); break;
case '-': upitem(); break;
case ',': prevpot(); break;
case '.': nxtpot(); break;
case 'Z': fpoint=scan(1);
        fstart=fpoint;
        clear_window(wnd_filter);
        helpfunc=help;
        break;
case 'R': for (j=0; j<fcnt1+1; j++) {
            if (filter1[j].tag == 1) filter1[j].tag = 0;
            else filter1[j].tag = 1;
        }
        break;
case ALT_M: if (h + nfltags > nhold) break;
        for (j=0; j<fcnt1+1; j++) {

```

```

        if (filter1[j].tag == 1) {
            ++h;
            strcpy(hold[h].name,filter1[j].name);
            strcpy(hold[h].id,filter1[j].id);
            strcpy(hold[h].potency,filter1[j].potency);
            hold[h].potnum[0] = filter1[j].potnum[0];
            hold[h].potnum[1] = filter1[j].potnum[1];
            hold[h].potnum[2] = filter1[j].potnum[2];
            hold[h].num0 = filter1[j].num0;
            hold[h].num1 = filter1[j].num1;
            hold[h].num2 = filter1[j].num2;
            hold[h].num3 = filter1[j].num3;
            hold[h].num4 = filter1[j].num4;
            hold[h].num5 = filter1[j].num5;
            hold[h].num6 = filter1[j].num6;
            hold[h].num7 = filter1[j].num7;
            hold[h].num8 = filter1[j].num8;
            hold[h].num9 = filter1[j].num9;
            hold[h].max = max;
            hold[h].min = min;
            hold[h].rise = rise;
            hold[h].fall = fall;
            hold[h].tag = 1;
        }
    }
    twobells();
    writeholdfile();
    break;
case DEL: if (fcnt1 < 0) break;
    for (j=fpoint; j<fcnt1+1; j++) {
        strcpy(filter1[j].name,filter1[j+1].name);
        strcpy(filter1[j].potency,filter1[j+1].potency);
        filter1[j].potnum[0] = filter1[j+1].potnum[0];
        filter1[j].potnum[1] = filter1[j+1].potnum[1];
        filter1[j].potnum[2] = filter1[j+1].potnum[2];
        filter1[j].num1 = filter1[j+1].num1;
        filter1[j].num2 = filter1[j+1].num2;
        filter1[j].num3 = filter1[j+1].num3;
        filter1[j].num4 = filter1[j+1].num4;
        filter1[j].num5 = filter1[j+1].num5;
        filter1[j].tag = filter1[j+1].tag;
    }
    fcnt1--;
    beep();
    clear_window(wnd_filter);

```

```

        if (fpoint > fcnt1) {
            fpoint--;
            fstart = fpoint-19;
            if (fstart<0) fstart=0;
        }
        writefilter1();
        break;
case INS: if (fcnt1 < nfilter) {
    fcnt1++;
    for (j=fcnt1; j>fpoint-1; j--) {
        strcpy(filter1[j+1].name,filter1[j].name);
        strcpy(filter1[j+1].id,filter1[j].id);
        strcpy(filter1[j+1].potency,filter1[j].potency);
        filter1[j+1].potnum[0] = filter1[j].potnum[0];
        filter1[j+1].potnum[1] = filter1[j].potnum[1];
        filter1[j+1].potnum[2] = filter1[j].potnum[2];
        filter1[j+1].num0 = filter1[j].num0;
        filter1[j+1].num1 = filter1[j].num1;
        filter1[j+1].num2 = filter1[j].num2;
        filter1[j+1].num3 = filter1[j].num3;
        filter1[j+1].num4 = filter1[j].num4;
        filter1[j+1].num5 = filter1[j].num5;
        filter1[j+1].num6 = filter1[j].num6;
        filter1[j+1].num7 = filter1[j].num7;
        filter1[j+1].num8 = filter1[j].num8;
        filter1[j+1].num9 = filter1[j].num9;
        filter1[j+1].tag = filter1[j].tag;
    }
    strcpy(filter1[fpoint].name,outname);
    strcpy(filter1[fpoint].id,outid);
    strcpy(filter1[fpoint].potency,pot[dilindex].name);
    filter1[fpoint].potnum[0] = cPotNum[0];
    filter1[fpoint].potnum[1] = cPotNum[1];
    filter1[fpoint].potnum[2] = cPotNum[2];
    filter1[fpoint].num0 = outnum0;
    filter1[fpoint].num1 = outnum1;
    filter1[fpoint].num2 = outnum2;
    filter1[fpoint].num3 = outnum3;
    filter1[fpoint].num4 = outnum4;
    filter1[fpoint].num5 = outnum5;
    filter1[fpoint].num6 = outnum6;
    filter1[fpoint].num7 = outnum7;
    filter1[fpoint].num8 = outnum8;
    filter1[fpoint].num9 = outnum9;
    filter1[fpoint].tag = filter1[j].tag;
}

```

```

        beep();
        clear_window(wnd_filter);
        if (fpoint > fcnt1) {
            fpoint--;
            fstart = fpoint-19;
            if (fstart<0) fstart=0;
        }
        writefilter1();
    }
    break;
case 'P': prtstatus=biosprint(2,0,0);
    if (prtstatus==0x90) {
        printclient('N');
        showline("msg_Filter1Sel ");
        sprintf(string,msg_line);
        toprinter(string);
        printcrlf();
        for (j=0; j<fcnt1+1; j++) {
            if (filter1[j].tag) {
                sprintf(string,"%s %s",
                    filter1[j].name,filter1[j].potency);
                toprinter(string);
                printcrlf();
            }
        }
    }
    break;
default: break;
}

}

delete_window(wnd_filter);
filteroutflag=0;
fromkey='I';
}

void countf2tags()
{
    int i;
    nf2tags=0;
    for (i=0; i<fcnt2+1; i++) if (filter2[i].tag) nf2tags++;
}

void f2untags()
{
    int i;

```

```

for (i=0; i<nfilter+1; i++) filter2[i].tag = 0;
}

void allf2tags()
{
int i;
for (i=0; i<nfilter+1; i++) filter2[i].tag = 1;
}

void viewfilter2()
{
int j,temp,key=0;
int fpoint,fstart,end;
int prtstatus;
char string[50];
WINDOW *wnd_filter;

setmenuhelp("filter ",0,0);
wnd_filter = establish_window(39,0,25,41);
showline("msg_Filter2Sel ");
set_title(wnd_filter,msg_line);
set_colors(wnd_filter,ALL,BROWN,BLACK,DIM);
set_colors(wnd_filter,ACCENT,accentcolor,BROWN,BRIGHT);
display_window(wnd_filter);
fpoint=0;
fstart=0;
filteroutflag=1;
fromkey='2';
while (key != ESC) {
    wcursor(wnd_filter,0,0);
    if (fstart+20 <= fcnt2) end = fstart+20;
    else end = fcnt2+1;
    for (j=fstart; j<end; j++) {
        if (j == fpoint) reverse_video(wnd_filter);
        if (filter2[j].tag == 1) temp=16;
        else temp = ' ';
        wputchar(wnd_filter,temp);
        wprintf(wnd_filter,"%#2d.%s%s",
            j+1,filter2[j].name,filter2[j].potency);
        if (end != j) wprintf(wnd_filter,"\n");
        normal_video(wnd_filter);
    }
    countf2tags();
    wcursor(wnd_filter,0,22);
    showline("msg_FiltersTL ");
}

```

```

wprintf(wnd_filter,msg_line,fcnt2+1,nf2tags,nfilter+1);
key = get_char();
key = toupper(key);
switch(key)
{
case ' ': if (filter2[fpoint].tag == 1) filter2[fpoint].tag=0;
          else filter2[fpoint].tag = 1;
case RSWITCH:
case DN: fpoint++;
        if (fpoint > fcnt2) {
            fpoint=0;
            fstart=0;
        }
        if (fpoint > fstart+19) {
            fstart = fpoint;
            clear_window(wnd_filter);
        }
        break;
case LSWITCH:
case UP: fpoint--;
        if (fpoint < 0) {
            fpoint=fcnt2;
            fstart=fcnt2-19;
            if (fstart < 0) fstart=0;
        }
        if (fpoint < fstart) {
            fstart=fpoint-19;
            clear_window(wnd_filter);
            if (fstart < 0) {
                fstart=0;
                fpoint=0;
            }
        }
        break;
case PGUP: fstart=fstart-20;
        if (fstart < 0) fstart=0;
        clear_window(wnd_filter);
        fpoint=fstart;
        break;
case PGDN: fstart = fstart+20;
        if (fstart > fcnt2) fstart=0;
        clear_window(wnd_filter);
        fpoint=fstart;
        break;
case HOME: fstart=0;

```

```

        fpoint=0;
        break;
case END: fstart=fcnt2-19;
        if (fstart < 0) fstart=0;
        fpoint=fcnt2;
        break;
case 'W': display_time(); break;
case 'U': f2untags(); break;
case 'T': allf2tags(); break;
case '+': downitem(); break;
case '-': upitem(); break;
case ',': prevpot(); break;
case '!': nxtpot(); break;
case 'Z': fpoint=scan(1);
        fstart=fpoint;
        clear_window(wnd_filter);
        helpfunc=help;
        break;
case 'R': for (j=0; j<fcnt2+1; j++) {
        if (filter2[j].tag == 1) filter2[j].tag = 0;
        else filter2[j].tag = 1;
        }
        break;
case ALT_M: if (h + nf2tags > nhold) break;
        for (j=0; j<fcnt2+1; j++) {
        if (filter2[j].tag == 1) {
        ++h;
        strcpy(hold[h].name,filter2[j].name);
        strcpy(hold[h].id,filter2[j].id);
        strcpy(hold[h].potency,filter2[j].potency);
        hold[h].potnum[0] = filter2[j].potnum[0];
        hold[h].potnum[1] = filter2[j].potnum[1];
        hold[h].potnum[2] = filter2[j].potnum[2];
        hold[h].num0 = filter2[j].num0;
        hold[h].num1 = filter2[j].num1;
        hold[h].num2 = filter2[j].num2;
        hold[h].num3 = filter2[j].num3;
        hold[h].num4 = filter2[j].num4;
        hold[h].num5 = filter2[j].num5;
        hold[h].num6 = filter2[j].num6;
        hold[h].num7 = filter2[j].num7;
        hold[h].num8 = filter2[j].num8;
        hold[h].num9 = filter2[j].num9;
        hold[h].max = max;
        hold[h].min = min;

```

```

        hold[h].rise = rise;
        hold[h].fall = fall;
        hold[h].tag = 1;
    }

    }
    twobells();
    writeholdfile();
    break;
case DEL: if (fcnt2 < 0) break;
    for (j=fpoint; j<fcnt2+1; j++) {
        strcpy(filter2[j].name,filter2[j+1].name);
        strcpy(filter2[j].potency,filter2[j+1].potency);
        filter2[j].potnum[0] = filter2[j+1].potnum[0];
        filter2[j].potnum[1] = filter2[j+1].potnum[1];
        filter2[j].potnum[2] = filter2[j+1].potnum[2];
        filter2[j].num0 = filter2[j+1].num0;
        filter2[j].num1 = filter2[j+1].num1;
        filter2[j].num2 = filter2[j+1].num2;
        filter2[j].num3 = filter2[j+1].num3;
        filter2[j].num4 = filter2[j+1].num4;
        filter2[j].num5 = filter2[j+1].num5;
        filter2[j].num6 = filter2[j+1].num6;
        filter2[j].num7 = filter2[j+1].num7;
        filter2[j].num8 = filter2[j+1].num8;
        filter2[j].num9 = filter2[j+1].num9;
        filter2[j].tag = filter2[j+1].tag;
    }
    fcnt2--;
    beep();
    clear_window(wnd_filter);
    if (fpoint > fcnt2) {
        fpoint--;
        fstart = fpoint-19;
        if (fstart<0) fstart=0;
    }
    writefilter2();
    break;
case INS: if (fcnt2 < nfilter) {
    fcnt2++;
    for (j=fcnt2; j>fpoint-1; j--) {
        strcpy(filter2[j+1].name,filter2[j].name);
        strcpy(filter2[j+1].potency,filter2[j].potency);
        filter2[j+1].potnum[0] = filter2[j].potnum[0];
        filter2[j+1].potnum[1] = filter2[j].potnum[1];
        filter2[j+1].potnum[2] = filter2[j].potnum[2];
    }
}

```



```

        filter2[j+1].num0 = filter2[j].num0;
        filter2[j+1].num1 = filter2[j].num1;
        filter2[j+1].num2 = filter2[j].num2;
        filter2[j+1].num3 = filter2[j].num3;
        filter2[j+1].num4 = filter2[j].num4;
        filter2[j+1].num5 = filter2[j].num5;
        filter2[j+1].num6 = filter2[j].num6;
        filter2[j+1].num7 = filter2[j].num7;
        filter2[j+1].num8 = filter2[j].num8;
        filter2[j+1].num9 = filter2[j].num9;
        filter2[j+1].tag = filter2[j].tag;
    }
    strcpy(filter2[fpoint].name,outname);
    strcpy(filter2[fpoint].potency,pot[dilindex].name);
    filter2[fpoint].potnum[0] = cPotNum[0];
    filter2[fpoint].potnum[1] = cPotNum[1];
    filter2[fpoint].potnum[2] = cPotNum[2];
    filter2[fpoint].num0 = outnum0;
    filter2[fpoint].num1 = outnum1;
    filter2[fpoint].num2 = outnum2;
    filter2[fpoint].num3 = outnum3;
    filter2[fpoint].num4 = outnum4;
    filter2[fpoint].num5 = outnum5;
    filter2[fpoint].num6 = outnum6;
    filter2[fpoint].num7 = outnum7;
    filter2[fpoint].num8 = outnum8;
    filter2[fpoint].num9 = outnum9;
    filter2[fpoint].tag = filter1[j].tag;
    beep();
    clear_window(wnd_filter);
    if (fpoint > fcnt2) {
        fpoint--;
        fstart = fpoint-19;
        if (fstart<0) fstart=0;
    }
    writefilter2();
}

break;
case 'P': prtstatus=biosprint(2,0,0);
    if (prtstatus==0x90) {
        printclient('N');
        showline("msg_Filter2Sel ");
        sprintf(string,msg_line);
        toprinter(string);
        printf("\n");
    }

```

```

        for (j=0; j<fcnt2+1; j++) {
            if (filter2[j].tag) {
                sprintf(string,"%s %s",
                    filter2[j].name,filter2[j].potency);
                toprinter(string);
                printf("\n");
            }
        }
        break;
    default: break;
}

delete_window(wnd_filter);
filteroutflag=0;
fromkey='I';
}

/* REFERENCE INFO */

void reference()
{
    int key=0;
    setmenuhelp("refer ",0,0);
    while (key != ESC) {
        key=operationinfo("ReferIn",20,0,iskey);
        key=toupper(key);
        switch(key) {
            case 'K': pictures(); key=ESC; break;
            case 'C': organclock(); break;
            case 'R': reflexology(); break;
            case 'D': dental(); break;
            case 'Q': quinary(); break;
            default: break;
        }
    }
}

```

```
/* CLIENT Module for LISTEN Program */  
/* Copyright 1991-8 by James Hoyt Clark */
```

```
#include <dos.h>  
#include <stdio.h>  
#include <conio.h>  
#include <ctype.h>  
#include <stdlib.h>  
#include <alloc.h>  
#include <mem.h>  
#include <time.h>  
#include <string.h>  
#include <fcntl.h>  
#include <sys\types.h>  
#include <sys\stat.h>  
#include "twindow.h"  
#include "keys.h"
```

```
extern struct arearecord {  
    char name[31];  
    char id[8];  
    unsigned char num0;  
    unsigned char num1;  
    unsigned char num2;  
    unsigned char num3;  
    unsigned char num4;  
    unsigned char num5;  
    unsigned char num6;  
    unsigned char num7;  
    unsigned char num8;  
    unsigned char num9;  
    char select;  
    char biblio;  
    int position;  
};
```

```
extern struct arearecord area[areasize];
```

```
extern struct holdrecord {  
    char name[31];  
    char id[8];  
    unsigned char num0;  
    unsigned char num1;  
    unsigned char num2;  
    unsigned char num3;  
    unsigned char num4;
```

```

    unsigned char num5;
    unsigned char num6;
    unsigned char num7;
    unsigned char num8;
    unsigned char num9;
    unsigned char potency[11];
    unsigned char potnum[3];
    char max;
    char min;
    char rise;
    char fall;
    unsigned char tag;
};
extern struct holdrecord hold[90];

extern struct reportrecord {
    char name[31];
    char id[8];
    unsigned char potency[11];
    unsigned char max;
    unsigned char min;
    unsigned char rise;
    unsigned char fall;
};
extern struct reportrecord report[170];

extern struct pointrecord
{
    char ptname[8];
    unsigned char premax;
    unsigned char premin;
    unsigned char prerise;
    unsigned char prefall;
    unsigned char postmax;
    unsigned char postmin;
    unsigned char postrise;
    unsigned char postfall;
    char pcode;
    char arrow;
};
extern struct pointrecord point[1200];

extern struct pulldown {
    char line[26];
};

```

```
extern struct pulldown pulldn[25];
```

```
extern struct potrecord  
{  
    char name[8];  
    char num[6];  
    char CR;  
    char LF;  
};  
extern struct potrecord pot[1000];
```

```
extern int helpkey;  
extern char whichDRIVE;  
extern char h;  
extern int reportpoint;  
extern char clientdrive;  
extern long int npoint;  
extern char pointflag;  
extern char outholdflag;  
extern char pnum;  
extern char name[31];  
extern char id[8];  
extern unsigned int nitem;  
extern unsigned long int itempoint;  
extern unsigned char diskbuffer[100];  
extern int npot;  
extern char potname[11];  
extern int defaultvalues[100];  
extern int letter2color;  
extern int areapoint;  
extern int narea;  
extern char fromkey;  
extern int accentcolor;  
extern int color_ask;  
extern int color_delay;  
extern char govern;  
extern int pageline;  
extern int g_mode;  
extern WINDOW *wnd_pot;  
extern WINDOW *wnd_infokeep;  
extern char msg_line[80];
```

```
struct {  
    char last[20];  
    char first[20];
```

```

        char number[8];
        char nick[20];
        char addr[20];
        char city[20];
        char state[5];
        char zip[8];
        char ctry[20];
        char ethnic[11];
        char dt[7];
        char birth[7];
        char doc[9];
        char type[5];
        char sex[2];
        char phone[21];
        char amt[5];
        char extra[8];
        char visits;
        char samples;
    } client;

    struct {
        int systolic;
        int diastolic;
        int htrate;
        int temp;
        int weight;
        int height;
        int extra[25];
    } clientmeasure;

    struct hrecord {
        char name[31];
        char id[8];
        unsigned char num0;
        unsigned char num1;
        unsigned char num2;
        unsigned char num3;
        unsigned char num4;
        unsigned char num5;
        unsigned char num6;
        unsigned char num7;
        unsigned char num8;
        unsigned char num9;
        unsigned char potency[6];
        unsigned char potnum;
    }

```

```

    char max;
    char min;
    char rise;
    char fall;
    unsigned char tag;
};
struct hrecord holdold[90];

#define FIELDCHAR ' '
#define MAX_CHR 220
#define HARD 1
#define FLOP 0
#define LWID 60
#define WHT 10
#define PADHT 20

char bf [PADHT] [LWID];
char msk1 [] = " _ ";
char msk4 [] = " _ _ _ ";
char msk7 [] = " _ _ _ _ _ ";
char msk8 [] = " _ _ _ _ _ ";
char msk19 [] = " _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ ";
char msk21 [] = " _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ ";
char mskdate [] = " _ _ / _ _ / _ _ ";

char clientbuffer[MAX_CHR];
char formnumber[40];
char filename[90];
char wholenam[60];
char command[90];
int insert_mode = FALSE;    /* insert mode, TRUE/FALSE */
char savefind[31];
long int qfindnum;
void addfield(WINDOW *wnd, FIELD *fld);
void disp_field(WINDOW *wnd, char *bf, char *msk);
void data_value(WINDOW *wnd, FIELD *fld);
void insert_status(void);
void zeropoint();
int read_field(WINDOW *wnd, FIELD *fld);
void right_justify(char *s);
void right_justify_zero_fill(char *s);
int endstroke(int c);
int spaces(char *c);
int first;
int nclient;

```

```
int clientpoint=0;
long int allsize;
int allfilecount;
char visitchar[3];
WINDOW *wnd_client, *wnd_print, *wnd_find;
WINDOW *wnd_prod;
WINDOW *wnd_namelist;
WINDOW *wnd_who;
```

```
FIELD *fld, *fl;
char findname[31];
char ifclient;
```

```
void docommand()
{
    strcat(command,">nul");
    //printf("\n%s\n",command);
    system(command);
}
```

```
long get_disk_size(char drive)
{
    union REGS ireg, oreg;
    ireg.h.ah=0x36;
    ireg.h.dl = drive - 'A' + 1;
    intdos(&ireg,&oreg);
    return (long) ((long) oreg.x.dx * (long) oreg.x.ax * (long) oreg.x.cx);
}
```

```
long get_disk_free(char drive)
{
    union REGS ireg, oreg;
    ireg.h.ah=0x36;
    ireg.h.dl = drive - 'A' + 1;
    intdos(&ireg,&oreg);
    return (long) ((long) oreg.x.ax * (long) oreg.x.bx * (long) oreg.x.cx);
}
```

```
void formatnumber()
{
    int i;
    for (i=0; i<7; i++) if (client.number[i] == ' ') break;
    strncpy(formnumber,client.number,i);
    strcat(formnumber, ".DAT");
}
```



```

void formpath(char letter,int visitnumber,int which)
{
int i;
strcpy(filename,"C:\\WORK\\CLIENT\\X");
filename[15]=letter;
for (i=0; i<7; i++) if (client.number[i] == ' ') break;
strncat(filename,client.number,i);
strcat(filename,".");
if (which) {
    itoa(visitnumber,visitchar,10);
    strcat(filename,visitchar);
}
else strcat(filename,"*");
}

```

```

void getbuffer()
{
int i,j;
strncpy(client.last,clientbuffer,20);
j=0;
for (i=20; i<40; i++) {
    client.first[j]=clientbuffer[i];
    j++;
}
j=0;
for (i=40; i<60; i++) {
    client.nick[j]=clientbuffer[i];
    j++;
}
j=0;
for (i=60; i<68; i++) {
    client.number[j]=clientbuffer[i];
    j++;
}
j=0;
for (i=68; i<88; i++) {
    client.addr[j]=clientbuffer[i];
    j++;
}
j=0;
for (i=88; i<108; i++) {
    client.city[j]=clientbuffer[i];
    j++;
}
j=0;

```

```

for (i=108; i<113; i++) {
    client.state[j]=clientbuffer[i];
    j++;
}
j=0;
for (i=113; i<121; i++) {
    client.zip[j]=clientbuffer[i];
    j++;
}
j=0;
for (i=121; i<141; i++) {
    client.ctrj[j]=clientbuffer[i];
    j++;
}
j=0;
for (i=141; i<152; i++) {
    client.ethnic[j]=clientbuffer[i];
    j++;
}
j=0;
for (i=152; i<159; i++) {
    client.dt[j]=clientbuffer[i];
    j++;
}
j=0;
for (i=159; i<166; i++) {
    client.birth[j]=clientbuffer[i];
    j++;
}
j=0;
for (i=166; i<175; i++) {
    client.doc[j]=clientbuffer[i];
    j++;
}
j=0;
for (i=175; i<180; i++) {
    client.type[j]=clientbuffer[i];
    j++;
}
client.sex[0]=clientbuffer[180];
j=0;
for (i=182; i<203; i++) {
    client.phone[j]=clientbuffer[i];
    j++;
}

```

```

j=0;
for (i=203; i<208; i++) {
    client.amt[j]=clientbuffer[i];
    j++;
}
client.visits=clientbuffer[216];
client.samples=clientbuffer[217];
}

```

```

drivestatus()
{
#define DISK_READ 2
unsigned status;
char buf[512];
int retry;
int Drive;
int head=0;
int track=0;
int sector=8;
int nsectors=1;
if (clientdrive == 'A') Drive=0;
else Drive=1;
for (retry=0; retry<3; retry++)
    status=biosdisk(DISK_READ,Drive,head,track,sector,nsectors,buf);
return(status);
}

```

```

void savebuffer()
{
int i,j;
strcpy(clientbuffer,client.last);
j=0;
for (i=20; i<40; i++) {
    clientbuffer[i]=client.first[j];
    j++;
}
j=0;
for (i=40; i<60; i++) {
    clientbuffer[i]=client.nick[j];
    j++;
}
j=0;
for (i=60; i<68; i++) {
    clientbuffer[i]=client.number[j];
    j++;
}
}

```

```

    }
j=0;
for (i=68; i<88; i++) {
    clientbuffer[i]=client.addr[j];
    j++;
}
j=0;
for (i=88; i<108; i++) {
    clientbuffer[i]=client.city[j];
    j++;
}
j=0;
for (i=108; i<113; i++) {
    clientbuffer[i]=client.state[j];
    j++;
}
j=0;
for (i=113; i<121; i++) {
    clientbuffer[i]=client.zip[j];
    j++;
}
j=0;
for (i=121; i<141; i++) {
    clientbuffer[i]=client.ctry[j];
    j++;
}
j=0;
for (i=141; i<152; i++) {
    clientbuffer[i]=client.ethnic[j];
    j++;
}
j=0;
for (i=152; i<159; i++) {
    clientbuffer[i]=client.dt[j];
    j++;
}
j=0;
for (i=159; i<166; i++) {
    clientbuffer[i]=client.birth[j];
    j++;
}
j=0;
for (i=166; i<175; i++) {
    clientbuffer[i]=client.doc[j];
    j++;
}

```

```

    }
    j=0;
    for (i=175; i<180; i++) {
        clientbuffer[i]=client.type[j];
        j++;
    }
    clientbuffer[180]=client.sex[0];
    j=0;
    for (i=182; i<203; i++) {
        clientbuffer[i]=client.phone[j];
        j++;
    }
    j=0;
    for (i=203; i<208; i++) {
        clientbuffer[i]=client.amt[j];
        j++;
    }
    clientbuffer[216]=client.visits;
    clientbuffer[217]=client.samples;
}

void readname(long int position)
{
    int fhandle;
    if ((fhandle=open("C:\\WORK\\CLIENT\\CLIENT.DAT",O_RDONLY|O_BINARY)) == -1) {
        operationinfo("errorop",10,5,iskey);
        return;
    }
    lseek(fhandle,MAX_CHR*position,SEEK_CUR);
    if ((read(fhandle,clientbuffer,MAX_CHR)) == -1) {
        operationinfo("errorfl",10,5,iskey);
        exit(1);
    }
    getbuffer();
    close(fhandle);
}

long sizeclientfiles(long int clientnumber, char only)
{
    int fhandle, key=0;
    int i,filecount=0;
    long int size=0;
    struct stat info;
    WINDOW *wnd_size;
    readname(clientnumber);

```

```

formpath('N',0,HARD);
if (access(filename,0) == 0) {
    fhandle=open(filename,O_WRONLY|O_BINARY);
    fstat(fhandle,&info);
    size = info.st_size;
    close(fhandle);
    filecount++;
}
for (i=0; i<client.visits; i++) {
    formpath('P',i,HARD);
    if (access(filename,0) == 0) {
        fhandle=open(filename,O_WRONLY|O_BINARY);
        fstat(fhandle,&info);
        size = size + info.st_size;
        close(fhandle);
        filecount++;
    }
}
for (i=0; i<client.visits; i++) {
    formpath('H',i,HARD);
    if (access(filename,0) == 0) {
        fhandle=open(filename,O_WRONLY|O_BINARY);
        fstat(fhandle,&info);
        size = size + info.st_size;
        close(fhandle);
        filecount++;
    }
}
for (i=0; i<client.visits; i++) {
    formpath('R',i,HARD);
    if (access(filename,0) == 0) {
        fhandle=open(filename,O_WRONLY|O_BINARY);
        fstat(fhandle,&info);
        size = size + info.st_size;
        close(fhandle);
        filecount++;
    }
}
size=size+220;
allsize=allsize+size;
allfilecount=allfilecount+filecount;
if (only) {
    wnd_size = establish_window(33,10,3,33);
    showline("msg_ClientDtSize");
    set_title(wnd_size,msg_line);
}

```

```

        set_colors(wnd_size,ALL,BLACK,GREEN,BRIGHT);
        display_window(wnd_size);
        showline("msg_Filesbytes ");
        wprintf(wnd_size,msg_line,filecount+1,size);
        while (key != ESC) key = get_char();
        delete_window(wnd_size);
    }
    return size;
}

```

/* PRINTING */

```

void outprinter(int count, int character)
{
    int i;
    for (i=0; i<count; i++) biosprint(0,character,pnum);
}

```

```

void printcrlf()
{
    biosprint(0,0x0D,pnum);
    biosprint(0,0x0A,pnum);
    outprinter(4,0x20);
}

```

```

void printcrlf2()
{
    printcrlf();
    printcrlf();
}

```

```

void toprinter(char str[70])
{
    int i,data;
    for (i=0; str[i] !='\0'; i++) {
        data = str[i];
        biosprint(0,data,pnum);
    }
}

```

```

void devicemsg(void)
{
    char str[80];
    if (govern == 0) {
        printcrlf();
    }
}

```

```

        showline("msg_invest1  ");
        strcpy(str,msg_line);
        strcat(str,"\r\n  ");
        showline("msg_invest2  ");
        strcat(str,msg_line);
        toprinter(str);
    }
}

```

```

void formfeed(void)
{
    devicemsg();
    biosprint(0,0x0c,pnum);
}

```

```

void printline(void)
{
    int i,data;
    outprinter(64,0x2D);
    printcrlf();
}

```

```

void printlinecr()
{
    printcrlf();
    printline();
}

```

```

unspace_name()
{
    int i,j,k;
    char reducefirst[20];
    char reducelast[20];
    char reducenick[20];
    i=18;
    while (client.first[i] == ' ') i--;
    j=18;
    while (client.last[j] == ' ') j--;
    k=18;
    while (client.nick[k] == ' ') k--;
    strcpy(reducefirst,client.first);
    strcpy(reducelast,client.last);
    strcpy(reducenick,client.nick);
    reducefirst[i+1]=0;
    reducelast[j+1]=0;
}

```



```

reducenick[k+1]=0;
strcpy(wholename,reducefirst);
strcat(wholename," ");
strcat(wholename,reducelast);
strcat(wholename," ");
strcat(wholename,reducenick);
return i+j+k+6;
}

```

```

void printclient(char namenum)

```

```

{
char str[70];
struct tm *tm_now;
char *str_now;
long secs_now;

toprinter(": ");
if (namenum == 'N') {
    unspace_name();
    strcpy(str,wholename);
}
else strcat(str,client.number);
toprinter(str);
printf("\n");
time(&secs_now);
tm_now=localtime(&secs_now);
str_now=asctime(tm_now);
str_now[24]=0;
strcpy(str,str_now);
toprinter(str);
outprinter(2,0x20);
showline("msg_Testedby ");
strcpy(str,msg_line);
strcat(str,client.doc);
toprinter(str);
printf("\n");
printf("\n");
printf("\n");
}

```

```

void clientlisthead()

```

```

{
char str[80];
printf("\n");
showline("msg_LISTENSysCL ");

```

```

sprintf(str,msg_line);
toprinter(str);
printf("\n");
showline("msg_LastNameOthr");
sprintf(str,msg_line);
toprinter(str);
printf("\n");
}

```

```

void printclientinfo()
{
    unsigned char prtstatus;
    char str[70];
    prtstatus = biosprint(2,0,pnum);
    if (prtstatus != 0x90) {
        tellprinter();
        return;
    }
    printf("\n");
    showline("msg_Number ");
    toprinter(msg_line);
    sprintf(str," %s",client.number);
    toprinter(str);
    printf("\n");
    showline("msg_Last ");
    toprinter(msg_line);
    sprintf(str," %s",client.last);
    toprinter(str);
    printf("\n");
    showline("msg_First ");
    toprinter(msg_line);
    sprintf(str," %s",client.first);
    toprinter(str);
    printf("\n");
    showline("msg_Other ");
    toprinter(msg_line);
    sprintf(str," %s",client.nick);
    toprinter(str);
    printf("\n");
    showline("msg_Address ");
    toprinter(msg_line);
    sprintf(str," %s",client.addr);
    toprinter(str);
    printf("\n");
    showline("msg_City ");
}

```

```

toprinter(msg_line);
sprintf(str, " %s", client.city);
toprinter(str);
printf("\n");
showline("msg_State    ");
toprinter(msg_line);
sprintf(str, " %s", client.state);
toprinter(str);
printf("\n");
showline("msg_Zip      ");
toprinter(msg_line);
sprintf(str, " %s", client.zip);
toprinter(str);
printf("\n");
showline("msg_Country   ");
toprinter(msg_line);
sprintf(str, " %s", client.ctr);
toprinter(str);
formfeed();
}

```

```

void printclients()
{
    unsigned char prtstatus;
    char str[70];
    char shortnick[16];
    int j;
    struct tm *tm_now;
    long secs_now;
    char *str_now;
    div_t result;
    prtstatus = biosprint(2, 0, pnum);
    if (prtstatus != 0x90) {
        tellprinter();
        return;
    }
    clientlisthead();
    for (j=0; j<nclient-1; j++) {
        readname(j);
        strncpy(shortnick, client.nick, 16);
        shortnick[15]=0;
        sprintf(str, "%#3d. %s %s %s %#3d %s", j+1, client.last, client.first,
            client.number, client.visits, shortnick);
        toprinter(str);
        if (j != 0) {

```

```

        result = div(j,53);
        if (result.rem == 0) {
            formfeed();
            clientlisthead();
        }
    }
    printf("\n");
}

printf("\n");
time(&secs_now);
tm_now = localtime(&secs_now);
str_now = asctime(tm_now);
str_now[24] = 0;
showline("msg_DateandTime ");
sprintf(str,msg_line,str_now);
toprinter(str);
formfeed();
}

void tellwho()
{
    int xlen;
    xlen=unspace_name();
    wnd_who = establish_window(9,3,3,xlen);
    set_colors(wnd_who,ALL,BLACK,WHITE,BRIGHT);
    set_colors(wnd_who,ACCENT,WHITE,BLACK,DIM);
    display_window(wnd_who);
    wprintf(wnd_who,"%s",wholename);
}

void whichnotefile()
{
    if (client.last[0] == 0) strcpy(filename,"C:\\WORK\\NOTES.DAT");
    else formpath('N',0,HARD);
}

void printnotes()
{
    FILE *fp, *fopen();
    int i,ctr = 0, data;
    char str[64];
    setmem(bf, sizeof bf, 0);
    whichnotefile();
    if ((fp = fopen(filename,"rt")) != NULL) {
        showline("msg_LISTENSystemNot");
    }
}

```

```

        strcpy(str,msg_line);
        toprinter(str);
        printclient('N');
        printcrlf2();
        while (fread(bf[ctr], LWID, 1, fp)) {
            strcpy(str,bf[ctr]);
            toprinter(str);
            ctr++;
            printcrlf();
        }
    }
    fclose(fp);
    formfeed();
}

void headline()
{
    char str[70];
    printcrlf();
    showline("msg_headline  ");
    strcpy(str,msg_line);
    toprinter(str);
    printlinecr();
}

void printhold()
{
    int j;
    char str[80];
    printcrlf();
    showline("msg_HoldReport ");
    strcpy(str,msg_line);
    toprinter(str);
    printclient('N');
    headline();
    for (j=0; j<h+1; j++) {
        sprintf(str,"%#2d. %s %s %#3d %#3d %#3d %#3d %s",
            j+1,hold[j].name,hold[j].id,
            hold[j].max,hold[j].min,hold[j].rise,hold[j].fall,
            hold[j].potency);
        /* hold[j].potency,holdv[j].ptname);*/
        toprinter(str);
        printcrlf();
    }
    defaultvalues[29]++;
}

```

```

writedefaults();
formfeed();
}

void printtag()
{
int j;
char str[70];
printf();
showline("msg_HoldTagReprt");
strcpy(str,msg_line);
toprinter(str);
printclient('N');
headline();
for (j=0; j<h+1; j++)
    if (hold[j].tag) {
        sprintf(str,"%#2d. %s %s %#3d %#3d %#3d %#3d %s",
            j+1,hold[j].name,hold[j].id,
            hold[j].max,hold[j].min,hold[j].rise,hold[j].fall,
            hold[j].potency);
        toprinter(str);
        printf();
    }
printf2();
}

void printlist()
{
long int j,jstart,jcount;
int i,start,count;
int k=0;
div_t x;
int c=0;
char str[40];
char str2[70];
if (fromkey == 'P') return;
c=operationinfo("PrtList",10,4,yeskey);
c=toupper(c);
if (c != 'P') {
    tellcanceled();
    return;
}
operationkeep("AnyStop",10,4);
printf();
if (fromkey == 'I') {

```

```

jstart = itempoint;
count = nitem-jstart;
i=1;
for (j=itempoint; j<itempoint+count; j++) {
    getproduct(j);
    if (id[0] == ' ' && j != itempoint) break;
    if (id[0] == ' ') sprintf(str2, " %s",name);
    else {
        sprintf(str2, "  %#3d. %s",i,name);
        i++;
    }
    toprinter(str2);
    printf("\n");
    if (k == pageline) {
        biosprint(0,0x0c,pnum);
        printf("\n");
        k=0;
    }
    else k++;
    if (kbhit()) break;
}
}
if (fromkey == 'A') {
    start = areapoint;
    count = narea-start;
    for (i=start; i<start+count; i++) {
        if (area[i].biblio == 'x') toprinter("  ");
        strcpy(str,area[i].name);
        toprinter(str);
        printf("\n");
        if (k == pageline) {
            biosprint(0,0x0c,pnum);
            printf("\n");
            k=0;
        }
        else k++;
        if (kbhit()) break;
    }
}
biosprint(0,0x0c,pnum);
delete_window(wnd_infokeep);
}

void printreport()
{

```

```

int k=5;
int j;
char str[80];
printf();
showline("msg_ItemReadings");
strcpy(str,msg_line);
toprinter(str);
printclient('N');
headline();
for (j=0; j<reportpoint+1; j++) {
    sprintf(str,"%#3d. %s %s %#3d %#3d %#3d %#3d %s",j+1,report[j].name,
        report[j].id,report[j].max,report[j].min,report[j].rise,report[j].fall,
        report[j].potency);
    toprinter(str);
    printf();
    if (k == pageline) {
        biosprint(0,0x0c,pnum);
        printf();
        k=0;
    }
    else k++;
    if (kbhit()) break;
}
formfeed();
}

```

```

void printpoints()
{
    long int j;
    int k=5;
    char str[60];
    printf();
    showline("msg_BasePostRep ");
    strcpy(str,msg_line);
    toprinter(str);
    printclient('N');
    printf();
    showline("msg_BasePostHead");
    strcpy(str,msg_line);
    toprinter(str);
    printlinecr();
    for (j=0; j<npoint; j++) {
        if (point[j].premax != 0) {
            sprintf(str,"%s %#3d %#3d %#3d %#3d",point[j].ptname,
                point[j].premax,point[j].premin,point[j].prerise,point[j].prefall);

```



```

        topainter(str);
        outprinter(8,0x20);
        sprintf(str,"%s %#3d %#3d %#3d %#3d",point[j].ptname,
        point[j].postmax,point[j].postmin,point[j].postrise,point[j].postfall);
        topainter(str);
        if (k == pageline) {
            biosprint(0,0x0c,pnum);
            k=0;
        }
        else k++;
        if (kbhit()) break;
        printf("\n");
    }
    formfeed();
}

void research()
{
    long int j;
    int k=5;
    char str[60];
    printf("\n");
    showline("msg_ResearchRept");
    strcpy(str,msg_line);
    topainter(str);
    printclient('R');
    showline("msg_BaseMMIDRF ");
    strcpy(str,msg_line);
    topainter(str);
    printf("\n");
    for (j=0; j<npoint; j++) {
        if (point[j].premax != 0) {
            sprintf(str,"%s %#3d %#3d %#3d %#3d %#3d",point[j].ptname,
            point[j].premax,point[j].premin,point[j].premax-point[j].premin,
            point[j].prerise,point[j].prefall);
            topainter(str);
            if (k == pageline) {
                biosprint(0,0x0c,pnum);
                k=0;
            }
            else k++;
            if (kbhit()) break;
            printf("\n");
        }
    }
}

```

```

    }
formfeed();
}

```

```

void basebar()
{
long int j;
int i,k=5;
char str[60];
barheading(0);
for (j=0; j<npoint; j++) {
    if (point[j].premax != 0) {
        sprintf(str,"%s %#3d %#3d %#3d %#3d ",point[j].ptname,
            point[j].premax,point[j].premin,point[j].prerise,point[j].prefall);
        toprinter(str);
        for (i=0; i<(point[j].premin/2); i++) {
            if (i==24) biosprint(0,"|",pnum);
            else biosprint(0,".",pnum);
        }
        for (i=(point[j].premin/2); i<(point[j].premax/2); i++) {
            if (i==24) biosprint(0,"|",pnum);
            else biosprint(0,"*",pnum);
        }
        if (i<23) {
            for (i=i; i<24; i++) biosprint(0,' ',pnum);
            biosprint(0,"|",pnum);
        }
        if (k == pageline) {
            biosprint(0,0x0c,pnum);
            k=0;
        }
        else k++;
        if (kbhit()) break;
        printf("\n");
    }
}
formfeed();
}

```

```

void postbar()
{
unsigned int j;
int i,k=5;
char str[60];
barheading(1);

```

```

for (j=0; j<npoint; j++) {
    if (point[j].postmax != 0) {
        sprintf(str, "%s %#3d %#3d %#3d %#3d ", point[j].ptname,
            point[j].postmax, point[j].postmin, point[j].postrise, point[j].postfall);
        toprinter(str);
        for (i=0; i<(point[j].postmin/2); i++) {
            if (i==24) biosprint(0, '|', pnum);
            else biosprint(0, '.', pnum);
        }
        for (i=(point[j].postmin/2); i<(point[j].postmax/2); i++) {
            if (i==24) biosprint(0, '|', pnum);
            else biosprint(0, '*', pnum);
        }
        if (i<23) {
            for (i=i; i<24; i++) biosprint(0, ' ', pnum);
            biosprint(0, '|', pnum);
        }
        if (k == pageline) {
            biosprint(0, 0x0c, pnum);
            k=0;
        }
        else k++;
        if (kbhit()) break;
        printf("\n");
    }
}
formfeed();
}

```

```

char *rselcs[] = {
    pulldn[13].line, /*Hold*/
    pulldn[14].line, /*Hold Tag*/
    pulldn[15].line, /*Points*/
    pulldn[16].line, /*Base Bar*/
    pulldn[17].line, /*Post Bar*/
    pulldn[18].line, /*Items*/
    pulldn[19].line, /*Research*/
    pulldn[20].line, /*Notes*/
    NULL
};

char *ltselcs[] = {
    pulldn[21].line, /*List*/
    NULL
};

static void (*rfuncs[])() = {printhold, printtag, printpoints, basebar, postbar,

```

```

        printreport,research,printnotes};
static void (*lfuncs[])() = {printlist};
static MENU prtr [] = {
    {pulldn[22].line,rslcs,rfuncs},/*REPORTS*/
    {pulldn[23].line,lslcs,lfuncs},/*LISTS*/
    {NULL,NULL,NULL}
};

void printing()
{
    unsigned char prtstatus;
    prtstatus=biosprint(2,0,pnum);
    if (prtstatus==0x90) menu_print(prtr,1);
    else tellprinter();
}

/* NOTEPAD */

void notepad()
{
    WINDOW *wnd;
    FILE *fp, *fopen();
    int i, ctr = 0;
    tellwho();
    setmenuhelp("notepad",0,0);
    whichnotefile();
    setmem(bf, sizeof bf, ' ');
    if ((fp = fopen(filename,"rt")) != NULL) {
        while (fread(bf [ctr], LWID, 1, fp)) ctr++;
        fclose(fp);
    }

    wnd = establish_window((80-(LWID+2))/2, (25-(WHT+2))/2, WHT+2, LWID+2);
    set_border(wnd, 3);
    set_colors(wnd,ALL,WHITE,BLACK,DIM);
    set_colors(wnd,ACCENT,WHITE,BLACK,DIM);
    display_window(wnd);
    cursoron();
    text_editor(wnd, bf[0], (unsigned) LWID * PADHT);
    cursoroff();
    delete_window(wnd);
    ctr = PADHT;
    while (--ctr) {
        for (i = 0; i < LWID; i++)
            if (bf [ctr] [i] != ' ') break;
        if (i < LWID) break;
    }
}

```

```

    }
    fp = fopen(filename,"w");
    for (i = 0; i < ctr+1; i++)      fwrite(bf[i], LWID, 1, fp);
    fclose(fp);
    delete_window(wnd_who);
}

/* CLIENT */

/* ----- display a window prompt ----- */
void wprompt(WINDOW *wnd, int x, int y, char *s)
{
    wcursor(wnd, x, y);
    wprintf(wnd, s);
}

/* ----- provide today's date ----- */
void help_date(bf)
char *bf;
{
    struct date dat;
    getdate(&dat);
    sprintf(bf, "%02d%02d%02d",dat.da_day, dat.da_mon, dat.da_year % 100);
}

void init_template(WINDOW *wnd)
{
    fld = FHEAD;
    while (fld) {
        fl = fld->fnxt;
        free(fld);
        fld = fl;
    }
    FHEAD = FTAIL = NULL;
}

FIELD *establish_field(wnd, cl, rw, msk, bf, ty)
WINDOW *wnd;
int rw;
int cl;
char *msk;
char *bf;
int ty;
{
    if ( (fld = malloc(sizeof(FIELD))) == NULL)      return NULL;

```

```

fld->fmask = msk;
fld->frow = rw;
fld->fcol = cl;
fld->fbuff = bf;
fld->ftype = ty;
fld->fprot = 0;
fld->fnxt = fld->fprv = NULL;
fld->fvalid = NULL;
fld->fhelpt = NULL;
fld->fhwin = NULL;
fld->flx = fld->fly = 0;
addfield(wnd, fld);
return fld;
}

```

```

static void addfield(WINDOW *wnd, FIELD *fld)
{
    if (FTAIL) {
        fld->fprv = FTAIL;
        FTAIL->fnxt = fld;
    }
    FTAIL = fld;
    if (!FHEAD) FHEAD = fld;
}

```

```

static void disp_field(WINDOW *wnd, char *bf, char *msk)
{
    while (*msk) {
        wputchar(wnd, *msk != FIELDCHAR ? *msk : *bf++);
        msk++;
    }
}

```

```

static void data_value(WINDOW *wnd, FIELD *fld)
{
    wcursor(wnd, fld->fcol, fld->frow);
    disp_field(wnd, fld->fbuff, fld->fmask);
}

```

```

void field_tally(WINDOW *wnd)
{
    fld = FHEAD;
    while (fld != NULL) {
        data_value(wnd, fld);
        fld = fld->fnxt;
    }
}

```

```

    }
}

```

```

void field_window(FIELD *fld, char *hwin, int x, int y)
{
    fld->fhwin=hwin;
    fld->flx = x;
    fld->fly = y;
}

```

```

void clear_template(WINDOW *wnd)
{
    char *bf, *msk;
    fld = FHEAD;
    while (fld != NULL) {
        bf = fld->fbuff;
        msk = fld->fmask;
        while (*msk) {
            if (*msk == FIELDCHAR) *bf++ = ' ';
            msk++;
        }
        fld = fld->fnxt;
    }
    field_tally(wnd_client);
}

```

```

void clientcount()
{
    long int j;
    for (j=0; j<5000; j++) {
        readname(j);
        if (client.zip[0] == 'z') break;
    }
    nclient=j;
}

```

```

void writename(long int position,int delflg)
{
    int fhandle;
    struct stat info;
    savebuffer();
    if ((fhandle=open("C:\\WORK\\CLIENT\\CLIENT.DAT",O_WRONLY|O_BINARY)) == -1) {
        operationinfo("errorfl",10,10,nokey);
        exit(1);
    }
}

```

```

lseek(fhandle,MAX_CHR*position,SEEK_CUR);
if ((write(fhandle,clientbuffer,MAX_CHR)) == -1) {
    operationinfo("errorwt",10,10,nokey);
    exit(1);
}
if (delflg) {
    fstat(fhandle,&info);
    info.st_size = info.st_size-MAX_CHR;
    chsize(fhandle,info.st_size);
}
close(fhandle);
}

insertname()
{
    long int i;
    int j;
    char savenumber[9];
    char insbuffer[MAX_CHR];
    if (client.last[0] == '|' || client.number[0] == '|') {
        operationinfo("noempty",10,10,nokey);
        return 1;
    }
    savebuffer();
    strcpy(savenumber,client.number);
    for (j=0; j<MAX_CHR; j++) insbuffer[j] = clientbuffer[j];
    for (i=0; i<nclient; i++) {
        readname(i);
        if (strcmp(savenumber,client.number) == 0 ) {
            for (j=0; j<MAX_CHR; j++) clientbuffer[j] = insbuffer[j];
            operationinfo("numexis",20,5,nokey);
            return 1;
        }
    }
    readname(nclient);
    writename(nclient+1,0);
    for (i=nclient-1; i>-1; i--) {
        readname(i);
        writename(i+1,0);
        for (j=0; j<67; j++) if (insbuffer[j] != clientbuffer[j]) break;
        if (insbuffer[j] > clientbuffer[j]) break;
    }
    i++;
    if (i==0 && j<40) i=0;
    for (j=0; j<MAX_CHR; j++) clientbuffer[j] = insbuffer[j];

```



```

getbuffer();
writename(i,0);
nclient++;
clientpoint=i;
return 0;
}

```

```

void shiftnames(void)
{
long int i;
for(i=clientpoint; i<nclient; i++) {
    readname(i+1);
    writename(i,0);
}
writename(i,1);
nclient--;
}

```

```

void deletename()
{
int key;
if (nclient == clientpoint) return;
cursoroff();
twobells();
key=operationinfo("DELETE ",2,10,yeskey);
if (key == 'Y') {
    if (client.visits != 0) {
        strcpy(command,"DEL ");
        formpath('?',0,FLOP);
        strcat(command,filename);
        docommand();
    }
    shiftnames();
}
/*if (clientpoint == nclient) clientpoint--;*/
}

```

```

void show_visitnumber()
{
wcursor(wnd_client,8,19);
wprintf(wnd_client,"%#3d",client.visits);
}

```

```

void opmenu()
{

```

```
setmenuhelp("client ",0,0);
operationkeep("ClieOps",0,1);
}
```

```
void showinfo()
{
readname(clientpoint);
wnd_client = establish_window(32,2,22,35);
showline("msg_ClientPage ");
set_title(wnd_client,msg_line);
set_colors(wnd_client,ALL,6,BLACK,DIM);
set_colors(wnd_client,ACCENT,WHITE,BLACK,DIM);
display_window(wnd_client);
showline("msg_Number ");
wprompt(wnd_client,1,1,msg_line);
showline("msg_Last ");
wprompt(wnd_client,1,3,msg_line);
showline("msg_First ");
wprompt(wnd_client,1,4,msg_line);
showline("msg_Other ");
wprompt(wnd_client,1,5,msg_line);
showline("msg_Address ");
wprompt(wnd_client,1,6,msg_line);
showline("msg_City ");
wprompt(wnd_client,1,7,msg_line);
showline("msg_State ");
wprompt(wnd_client,1,8,msg_line);
showline("msg_Zip ");
wprompt(wnd_client,15,8,msg_line);
showline("msg_Country ");
wprompt(wnd_client,1,9,msg_line);
showline("msg_Phone ");
wprompt(wnd_client,1,10,msg_line);
showline("msg_StartDate ");
wprompt(wnd_client,1,12,msg_line);
showline("msg_BirthDate ");
wprompt(wnd_client,1,13,msg_line);
showline("msg_Sex ");
wprompt(wnd_client,1,14,msg_line);
showline("msg_Ethnic ");
wprompt(wnd_client,1,15,msg_line);
showline("msg_User ");
wprompt(wnd_client,1,17,msg_line);
showline("msg_Type ");
wprompt(wnd_client,19,17,msg_line);
```

```

showline("msg_Visits  ");
wprompt(wnd_client,1,19,msg_line);
init_template(wnd_client);
fld = establish_field(wnd_client,9,1,msk7,client.number,'A');
field_window(fld,"number ",40,0);
fld = establish_field(wnd_client,7,3,msk19,client.last,'A');
field_window(fld,"name   ",40,1);
fld = establish_field(wnd_client,8,4,msk19,client.first,'A');
field_window(fld,"name   ",40,2);
fld = establish_field(wnd_client,8,5,msk19,client.nick,'A');
field_window(fld,"name   ",40,2);
fld = establish_field(wnd_client,10,6,msk19,client.addr,'A');
field_window(fld,"address ",40,5);
fld = establish_field(wnd_client,7,7,msk19,client.city,'A');
field_window(fld,"address ",40,6);
fld = establish_field(wnd_client,8,8,msk4,client.state,'A');
field_window(fld,"state   ",40,7);
fld = establish_field(wnd_client,20,8,msk7,client.zip,'A');
field_window(fld,"address ",40,7);
fld = establish_field(wnd_client,10,9,msk19,client.ctrtry,'A');
field_window(fld,"address ",40,8);
fld = establish_field(wnd_client,8,10,msk21,client.phone,'A');
field_window(fld,"phone   ",40,9);
fld = establish_field(wnd_client,13,12,mskdate,client.dt,'A');
field_window(fld,"date    ",40,10);
fld = establish_field(wnd_client,13,13,mskdate,client.birth,'A');
field_window(fld,"birth   ",40,11);
fld = establish_field(wnd_client,6,14,msk1,client.sex,'A');
field_window(fld,"sex     ",40,14);
fld = establish_field(wnd_client,9,15,msk7,client.ethnic,'A');
field_window(fld,"ethnic  ",40,14);
fld = establish_field(wnd_client,7,17,msk8,client.doc,'A');
field_window(fld,"user    ",40,16);
fld = establish_field(wnd_client,25,17,msk4,client.type,'A');
field_window(fld,"type    ",40,16);
show_visitnumber();
}

```

```

/*void showmeasure()
{
readmeasure(client.visits);
wnd_measure = establish_window(32,2,22,35);
showline("msg_ClinicData ");
set_title(wnd_client,msg_line);
set_colors(wnd_client,ALL,6,BLACK,DIM);

```

```

set_colors(wnd_client,ACCENT,WHITE,BLACK,DIM);
display_window(wnd_client);
showline("msg_BP      ");
wprompt(wnd_client,1,1,msg_line);
showline("msg_Height   ");
wprompt(wnd_client,1,2,msg_line);
showline("msg_Weight    ");
wprompt(wnd_client,1,3,msg_line);
showline("msg_Temp      ");
wprompt(wnd_client,1,4,msg_line);
init_template(wnd_client);
fld = establish_field(wnd_client,9,1,msk7,systolic,'A');
field_window(fld,"number ",40,0);
fld = establish_field(wnd_client,7,3,msk19,diastolic,'A');
field_window(fld,"name   ",40,1);
fld = establish_field(wnd_client,8,4,msk19,Height,'A');
field_window(fld,"name   ",40,2);
fld = establish_field(wnd_client,8,5,msk19,Weight,'A');
field_window(fld,"name   ",40,2);
fld = establish_field(wnd_client,10,6,msk19,Temp,'A');
field_window(fld,"address ",40,5);
fld = establish_field(wnd_client,7,7,msk19,client.city,'A');
field_window(fld,"address ",40,6);
fld = establish_field(wnd_client,8,8,msk4,client.state,'A');
field_window(fld,"state   ",40,7);
fld = establish_field(wnd_client,20,8,msk7,client.zip,'A');
field_window(fld,"address ",40,7);
fld = establish_field(wnd_client,10,9,msk19,client.ctrty,'A');
field_window(fld,"address ",40,8);
fld = establish_field(wnd_client,8,10,msk21,client.phone,'A');
field_window(fld,"phone   ",40,9);
fld = establish_field(wnd_client,13,12,mskdate,client.dt,'A');
field_window(fld,"date    ",40,10);
fld = establish_field(wnd_client,13,13,mskdate,client.birth,'A');
field_window(fld,"birth   ",40,11);
fld = establish_field(wnd_client,6,14,msk1,client.sex,'A');
field_window(fld,"sex     ",40,14);
fld = establish_field(wnd_client,9,15,msk7,client.ethnic,'A');
field_window(fld,"ethnic  ",40,14);
fld = establish_field(wnd_client,7,17,msk8,client.doc,'A');
field_window(fld,"user    ",40,16);
fld = establish_field(wnd_client,25,17,msk4,client.type,'A');
field_window(fld,"type    ",40,16);
show_visitnumber();
} */

```

```

void zeroclient()
{
    int i;
    for (i=0; i<MAX_CHR-10; i++) clientbuffer[i] = ' ';
    clientbuffer[19]=0;
    clientbuffer[39]=0;
    clientbuffer[59]=0;
    clientbuffer[67]=0;
    clientbuffer[87]=0;
    clientbuffer[107]=0;
    clientbuffer[112]=0;
    clientbuffer[120]=0;
    clientbuffer[140]=0;
    clientbuffer[151]=0;
    clientbuffer[158]=0;
    clientbuffer[165]=0;
    clientbuffer[174]=0;
    clientbuffer[179]=0;
    clientbuffer[181]=0;
    clientbuffer[201]=0;
    clientbuffer[207]=0;
    clientbuffer[216]=0;
    clientbuffer[217]=0;
    clientbuffer[218]=0x0d;
    clientbuffer[219]=0x0a;
    getbuffer();
}

```

```

void setfloppy()
{
    strcpy(filename,"A:\\");
    if (clientdrive=='B') filename[0]='B';
    formatnumber();
    strcat(filename,formnumber);
}

```

```

void clearfindname()
{
    int i;
    for (i=0; i<31; i++) findname[i] = 0;
}

```

```

void restorefromflop()
{
    WINDOW *wnd_restore;

```

```

struct ffbk ffbk;
int spaceplace,key;
int fhandle;
char store [7];
if (drivestatus() != 0) {
    notready();
    return;
}
strcpy(filename,"A:\\*.DAT");
if (clientdrive=='B') filename[0]='B';
findfirst(filename,&ffbk,0);
strcpy(filename,"A:\\");
if (clientdrive=='B') filename[0]='B';
strcat(filename,ffbk.ff_name);
if ((fhandle=open(filename,O_RDONLY)) == -1) {
    noclientinfo();
    return;
}
lseek(fhandle,0,SEEK_CUR);
if ((read(fhandle,clientbuffer,MAX_CHR)) == -1) {
    readerror();
    return;
}
getbuffer();
close(fhandle);
wnd_restore = establish_window(22,10,9,39);
set_colors(wnd_restore,ALL,BLACK,AQUA,DIM);
set_colors(wnd_restore,ACCENT,WHITE,BLACK,DIM);
display_window(wnd_restore);
wprintf(wnd_restore," %s",client.number);
wprintf(wnd_restore,"\n %s",client.last);
wprintf(wnd_restore,"\n %s",client.first);
wprintf(wnd_restore,"\n %s\n",client.nick);
showline("msg_ENTERclient ");
wprintf(wnd_restore,msg_line);
wprintf(wnd_restore,"\n");
showline("msg_Anyotherkey ");
wprintf(wnd_restore,msg_line);
key = get_char();
delete_window(wnd_restore);
if (key == ENTER) {
    if (!insertname()) {
        if (client.visits != 0) {
            wcursor(wnd_restore,7,3);
            operationinfo("Restorg",2,5,nokey);

```

```

        strcpy(command,"COPY A:\\?");
        if (clientdrive=='B') command[5]='B';
        spaceplace=strcspn(client.number, " ");
        strncat(command,client.number,spaceplace);
        strcat(command, ". *");
        strcat(command, " C:\\\\WORK\\CLIENT\\*. *");
        docommand();
        twobells();
    }
}

}

void readpoint(int number)
{
    FILE *fptr;
    npoint=0;
    formpath('P',number,HARD);
    if( (fptr=fopen(filename,"rb"))==NULL )    {
        operationinfo("errorfl",10,5,nokey);
        return;
    }
    else {
        while(fread(&point[npoint],sizeof(point[0]),1,fptr)==1) npoint++;
        fclose(fptr);
    }
}

void readvisit(int number)
{
    FILE *fptr;
    int key;
    int nhold;
    key=operationinfo("readvis",5,1,iskey);
    if (key == 'Y') {
        operationinfo("Retriev",5,1,nokey);
        formpath('N',0,HARD);
        if (access(filename,0) == 0) {
            strcpy(command,"COPY ");
            strcat(command,filename);
            strcat(command, " C:\\\\WORK\\NOTES.DAT /y");
            docommand();
        }
        h=0;
        nhold=0;
    }
}

```

```

formpath('H',number,HARD);
if (access(filename,0) != 0) {
    operationinfo("absentH",10,5,nokey);
}
else {
    fptr=fopen(filename,"rb");
    while(fread(&hold[h],sizeof(hold[0]),1,fptr)==1 )
    {
        if (hold[0].potency[5] == 0)
        {
            fclose(fptr);
            fptr=fopen(filename,"rb");
            while(fread(&holdold[nhold],sizeof(holdold[0]),1,fptr) == 1)
            {
                strcpy(hold[nhold].name,holdold[nhold].name);
                strcpy(hold[nhold].id,holdold[nhold].id);
                hold[nhold].num0 = holdold[nhold].num0;
                hold[nhold].num1 = holdold[nhold].num1;
                hold[nhold].num2 = holdold[nhold].num2;
                hold[nhold].num3 = holdold[nhold].num3;
                hold[nhold].num4 = holdold[nhold].num4;
                hold[nhold].num5 = holdold[nhold].num5;
                hold[nhold].num6 = holdold[nhold].num6;
                hold[nhold].num7 = holdold[nhold].num7;
                hold[nhold].num8 = holdold[nhold].num8;
                hold[nhold].num9 = holdold[nhold].num9;
                strcpy(hold[nhold].potency,holdold[nhold].potency);
                hold[nhold].potnum[2] = holdold[nhold].potnum;
                hold[nhold].potnum[0] = 0;
                hold[nhold].potnum[1] = 0;
                hold[nhold].max = holdold[nhold].max;
                hold[nhold].min = holdold[nhold].min;
                hold[nhold].rise = holdold[nhold].rise;
                hold[nhold].fall = holdold[nhold].fall;
                hold[nhold].tag = holdold[nhold].tag;
                nhold++;
            }
            beep();
            h = nhold;
            break;
        }
    }
    else
    {
        if (nhold) break;
        h++;
    }
}

```



```

        }
        fclose(fptr);
    }
    h--;
    reportpoint=0;
    formpath('R',number,HARD);
    if (access(filename,0) != 0) {
        operationinfo("absentR",10,5,nokey);
    }
    else {
        fptr=fopen(filename,"rb");
        while(fread(&report[reportpoint],sizeof(report[0]),1,fptr)==1 )
            reportpoint++;
        fclose(fptr);
    }
    reportpoint--;
    readpoint(number);
}
else tellcanceled();
}

```

```

void backuptoflop()
{
    FILE *infile;
    WINDOW *wnd_message;
    long int totalFDspace,freeFDspace,size;
    char name[10];
    int i,key = 'Y';
    if (client.visits == 0) return;
    if (drivestatus() != 0) {
        notready();
        return;
    }
    wnd_message = establish_window(14,1,19,49);
    showline("msg_BackupCheck ");
    set_title(wnd_message, msg_line);
    set_colors(wnd_message,ALL,RED,WHITE,DIM);
    display_window(wnd_message);
    totalFDspace=get_disk_size(clientdrive);
    freeFDspace=get_disk_free(clientdrive);
    size=sizeclientfiles(clientpoint,0);
    if (size > freeFDspace) {
        twobells();
        showline("msg_Toomuchdata ");
    }
}

```

```

    wprompt(wnd_message,1,2,msg_line);
    showline("msg_Pressanykey ");
    wprompt(wnd_message,1,2,msg_line);
    showline("msg_TotalSizeFD ");
    wprompt(wnd_message,1,2,msg_line);
    wprintf(wnd_message," %#7ld",totalFDspace);
    showline("msg_FreeSpaceFD ");
    wprompt(wnd_message,1,2,msg_line);
    wprintf(wnd_message," %#7ld",freeFDspace);
    showline("msg_SizeVisitDat");
    wprompt(wnd_message,1,2,msg_line);
    wprintf(wnd_message," %#7ld",size);
    get_char();
    delete_window(wnd_message);
    return;
}

delete_window(wnd_message);
key=operationinfo("BACKUPF",10,10,iskey);
if (key == 'Y') {
    setfloppy();
    if (access(filename,0) == 0) {
        twobells();
        key=operationinfo("BACKUPE",10,10,iskey);
    }
    if (key == 'Y') {
        operationinfo("BackupC",10,10,nokey);
//        printf("\nfil:%s", filename);
//        twobells();
//        getch();
        infile=fopen(filename,"wb");
        fwrite(clientbuffer,sizeof(char),MAX_CHR,infile);
        fflush(infile);
//        twobells();
//        printf("\ninfile");
//        getch();
        fclose(infile);
//        printf("\nfil=%s", filename);
//        twobells();
//        getch();
        strcpy(command,"COPY ");
        formpath('?',0,FLOP);
        strcat(command,filename);
//        printf("\nFLP=%s",command);
//        twobells();
//        getch();

```

```

        strcpy(filename, " A:\\*. *");
        if (clientdrive=='B') filename[1]='B';
        strcat(command,filename);
//      printf("\ncom=%s:", command);
//      twobells();
//      getch();
        docommand();
        twobells();
    }
}
else tellcanceled();
}

```

```

void visitdates()
{
    struct stat info;
    int i,row,col,start,adjust;
    int sel;
    int c1=0;
    WINDOW *wnd_vst;
    setmenuhelp("visitda",20,20);
    col=1+client.visits/21;
    wnd_vst = establish_window(5,0,25,31*(col));
    showline("msg_VisitDates ");
    set_title(wnd_vst,msg_line);
    set_colors(wnd_vst,ALL,BLUE,WHITE,DIM);
    set_colors(wnd_vst,ACCENT,WHITE,BLACK,DIM);
    display_window(wnd_vst);
    unspace_name();
    wprompt(wnd_vst,1,0,wholename);
    row=0;
    if (client.visits > 40) start = client.visits-40;
    else start=0;
    sel=start;
    while(c1 != ESC) {
        for (i=start; i<client.visits; i++) {
            formpath('P',i,HARD);
            if (sel == i) reverse_video(wnd_vst);
            if (stat(filename,&info) != 0) {
                printf("Error: %s",filename);
                return;
            }
            adjust=i-start;
            col = adjust/20;
            row = adjust - (20*col) + 2;

```

```

        wcursor(wnd_vst,col*30+1,row);
        wprintf(wnd_vst,"%#2d. ",i+1);
        wprintf(wnd_vst,"%s",ctime(&info.st_atime));
        normal_video(wnd_vst);
    }
    cursoroff();
    c1=get_char();
    c1=toupper(c1);
    switch(c1) {
        case UP: sel--;
                if (sel < start) sel=client.visits-1;
                break;
        case DN: sel++;
                if (sel > client.visits-1) sel=start;
                break;
        case BS:
        case FWD: sel=sel+20;
                if (sel > client.visits-1) sel=sel-40;
                break;
        case INS: readvisit(sel);
                c1=ESC;
                break;
        default:beep(); break;
    }
}
delete_window(wnd_vst);
}

void lastvisit()
{
    if (client.visits == 0) return;
    readvisit(client.visits-1);
    viewhold();
}

void firstvisit()
{
    if (client.visits == 0) return;
    readvisit(0);
    viewhold();
}

void archivevisit()
{
    int key;

```

```

key=operationinfo("Archive",10,5,iskey);
if (key != 'Y') {
    tellcanceled();
    return;
}
operationinfo("ArchivY",10,5,nokey);
strcpy(command,"COPY C:\\WORK\\PTREAD.DAT ");
formpath('P',client.visits,HARD);
strcat(command,filename);
docommand();
operationinfo("PointsA",10,5,nokey);
if (h > -1) {
    strcpy(command,"COPY C:\\WORK\\HOLD.DAT ");
    formpath('H',client.visits,HARD);
    strcat(command,filename);
    docommand();
    operationinfo("HoldArc",10,5,nokey);
}
if (reportpoint > -1) {
    strcpy(command,"COPY C:\\WORK\\RECORD.DAT ");
    formpath('R',client.visits,HARD);
    strcat(command,filename);
    docommand();
    operationinfo("RecordA",10,5,nokey);
}
client.visits++;
writename(clientpoint,0);
show_visitnumber();
delay(1000);
putchar(BELL);
}

void readirecfile()
{
FILE *fptr;
reportpoint=0;
if( (fptr=fopen("RECORD.DAT","rb"))==NULL ) {
    operationinfo("absentR",10,5,nokey);
}
else {
    while(fread(&report[reportpoint],sizeof(report[0]),1,fptr)==1 )
        reportpoint++;
    fclose(fptr);
}
reportpoint--;

```

```
}
```

```
void writeirecfile()
```

```
{
```

```
FILE *fptr;
```

```
int rcount=reportpoint+1;
```

```
if( (fptr=fopen("RECORD.DAT","wb"))==NULL ) {
```

```
    operationinfo("absentR",10,5,nokey);
```

```
}
```

```
else {
```

```
    while(fwrite(report,sizeof(report[0]),rcount,fptr)==1 ) rcount++;
```

```
    fclose(fptr);
```

```
}
```

```
}
```

```
void readpotfile()
```

```
{
```

```
FILE *fptr;
```

```
npot=0;
```

```
if( (fptr=fopen("POT.DAT","rb"))==NULL ) {
```

```
    operationinfo("errorfl",10,5,iskey);
```

```
    exit(0);
```

```
}
```

```
else {
```

```
    while(fread(&pot[npot],sizeof(pot[0]),1,fptr)==1) npot++;
```

```
    fclose(fptr);
```

```
}
```

```
}
```

```
void restorevisit()
```

```
{
```

```
int key;
```

```
twobells();
```

```
key=operationinfo("Restore",10,5,iskey);
```

```
switch (key) {
```

```
    case 'Y': twobells();
```

```
        operationinfo("RestorV",10,5,nokey);
```

```
        readholdfile();
```

```
        readirecfile();
```

```
        break;
```

```
    case 'd':
```

```
        case 'D': restoredefaults();
```

```
            operationinfo("DefRest",10,5,nokey);
```

```
            break;
```

```
    case ESC: tellcanceled();
```

```

        break;
    }
}

void checkroom()
{
    int key;
    long size;
    WINDOW *wnd_message;
    size=get_disk_free('C');
    if(size < 1000000) {
        wnd_message = establish_window(14,1, 10,40);
        set_colors(wnd_message,ALL,RED,WHITE,DIM);
        display_window(wnd_message);
        showline("msg_HardDriveFre");
        wprompt(wnd_message,1,1,msg_line);
        wprintf(wnd_message," %ld",size);
        showline("msg_Youless1MB ");
        wprompt(wnd_message,1,3,msg_line);
        showline("msg_ofspaceleft ");
        wprompt(wnd_message,1,4,msg_line);
        showline("msg_CLEARATAHD ");
        wprompt(wnd_message,1,6,msg_line);
        showline("msg_PressESCkey ");
        wprompt(wnd_message,1,8,msg_line);
        while (key != ESC) key=get_char();
        delete_window(wnd_message);
    }
}

```

```

static void insert_status()
{
    set_cursor_type(insert_mode ? 0x0106 : 0x0607);
}

```

/* CLIENT PAGE */

```

void display20client()
{
    long int first,row,j;
    first=clientpoint-10;
    row=0;
    if (clientpoint < 11 ) {
        row = 10-clientpoint;
        first = 0;
    }
}

```

```

    }
    wcursor(wnd_namelist,0,0);
    for (j=0; j<row; j++) {
        if (j != 0 ) wprintf(wnd_namelist, "\n");
        printspace(wnd_namelist,78);
    }
    for (j=first; j<clientpoint+13; j++) {
        if (row != 0) if ((clientpoint+13) != j) wprintf(wnd_namelist, "\n");
        row = 1;
        readname(j);
        if (clientpoint==j) reverse_video(wnd_namelist);
        if (j < nclient) wprintf(wnd_namelist, "%s %s %s  %#3d  %s",
                                client.last, client.first, client.number, client.visits, client.nick);
        else printspace(wnd_namelist,78);
        normal_video(wnd_namelist);
    }
}

```

```

void upclient()
{
    if (clientpoint == 0) clientpoint = nclient;
    --clientpoint;
}

```

```

void downclient()
{
    if (clientpoint > nclient-2) clientpoint = -1;
    ++clientpoint;
}

```

```

void clientlist()
{
    int i,c=0;
    setmenuhelp("cllist ",2,8);
    wnd_namelist = establish_window(0,0,25,80);
    showline("msg_PrtLastFirst");
    set_title(wnd_namelist,msg_line);
    set_colors(wnd_namelist,ALL,WHITE,RED,DIM);
    set_colors(wnd_namelist,ACCENT,BLACK,YELLOW,BRIGHT);
    display_window(wnd_namelist);
    while (c != ESC) {
        cursoroff();
        display20client();
        c = get_char();
        c = toupper(c);
    }
}

```



```

        if (c > '@' && c < '[') {
            for (i=0; i<nclient; i++) {
                readname(i);
                if (c == client.last[0]) {
                    clientpoint=i;
                    break;
                }
            }
        }
        switch(c) {
            case UP: upclient(); break;
            case DN: downclient(); break;
            case HOME: clientpoint=0;
                        break;
            case END: clientpoint = nclient-1;
                        break;
            case F4: clear_window(wnd_namelist);
                        wcursor(wnd_namelist,5,10);
                        wprintf(wnd_namelist,"Printing..");
                        printclients();
                        break;
            default: break;
        }
    }
    delete_window(wnd_namelist);
}

```

```

static int read_field(WINDOW *wnd, FIELD *fld)

```

```

{
    char *mask = fld->fmask, *buff = fld->fbuff;
    int done = FALSE, c, column;
    column = fld->fcol;
    while (*mask != FIELDCHAR) {
        column++;
        mask++;
    }
    while (TRUE) {
        wcursor(wnd, column, fld->frow);
        c = get_char();
        if (fld->ftype == 'A') c = toupper(c);
        clear_message();
        switch (c) {
            case '\b':
            case BS:
                if (buff == fld->fbuff) {

```

```

        done = c == BS;
        break;
    }
    --buff;
do    {
        --mask;
        --column;
    } while (*mask != FIELDCHAR);
if (c == BS)
    break;

case DEL:
    movmem(buff+1, buff, strlen(buff));
    *(buff+strlen(buff)) = 0;
    wcursor(wnd, column, fld->frow);
    disp_field(wnd, buff, mask);
    break;

case FWD:
    do    {
        column++;
        mask++;
    } while (*mask && *mask != FIELDCHAR);
    buff++;
    break;

case INS:
    insert_mode ^= TRUE;
    insert_status();
    break;

case ALT_P: printclient('N'); break;
case '!':
    if (fld->ftype == 'C') {
        if (*mask++ && *buff == ' ') {
            *buff++ = '0';
            if (*mask++ && *buff == ' ') *buff++ = '0';
        }
        right_justify(fld->fbuff);
        wcursor(wnd, fld->fcol, fld->frow);
        disp_field(wnd, fld->fbuff, fld->fmask);
        column = fld->fcol+strlen(fld->fmask)-2;
        mask = fld->fmask+strlen(fld->fmask)-2;
        buff = fld->fbuff+strlen(fld->fbuff)-2;
        break;
    }

default:
    if (endstroke(c)) {
        done = TRUE;
    }

```

```

        break;
    }
    if (toupper(fld->ftype) != 'A' && !isdigit(c)) {
        operationinfo("numonly", 10, 5, nokey);
        break;
    }
    if (insert_mode) {
        movmem(buff, buff+1, strlen(buff)-1);
        disp_field(wnd, buff, mask);
        wcursor(wnd, column, fld->frow);
    }
    *buff++ = c;
    wputchar(wnd, c);
    do {
        column++;
        mask++;
    } while (*mask && *mask != FIELDCHAR);
    if (!*mask) c = FWD;
    break;
}
if (!*mask) done = TRUE;
if (done) break;
}
if (c != ESC && toupper(fld->ftype) != 'A') {
    if (fld->ftype == 'C') {
        if (*mask++ && *buff == ' ') {
            *buff++ = '0';
            if (*mask++ && *buff == ' ')
                *buff++ = '0';
        }
    }
    if (fld->ftype == 'Z' || fld->ftype == 'D')
        right_justify_zero_fill(fld->fbuff);
    else
        right_justify(fld->fbuff);
    wcursor(wnd, fld->fcol, fld->frow);
    disp_field(wnd, fld->fbuff, fld->fmask);
}
return c;
}

```

```

static int endstroke(int c)
{
    switch (c) {
        case 'r':

```

```

    case '\n':
    case '\t':
    case ESC:
    case F1:
    case F2:
    case F3:
    case F4:
    case F5:
    case F6:
    case F7:
    case F8:
    case F9:
    case F10:
    case PGDN:
    case PGUP:
    case HOME:
    case END:
    case UP:
    case DN: return TRUE;
    default: return FALSE;
}

```

```

}

```

```

static void right_justify(char *s)
{
    int len;
    len = strlen(s);
    while (*s == ' ' || *s == '0' && len)    {
        len--;
        *s++ = ' ';
    }
    if (len)
        while (*(s+(len-1)) == ' ')    {
            movmem(s, s+1, len-1);
            *s = ' ';
        }
}

```

```

static void right_justify_zero_fill(char *s)
{
    int len;
    if (spaces(s)) return;
    len = strlen(s);
    while (*(s + len - 1) == ' ') {
        movmem(s, s + 1, len-1);
    }
}

```

```

        *s = '0';
    }
}

```

```

void newclient()
{
    readname(clientpoint);
    if (nclient != clientpoint) {
        field_tally(wnd_client);
        wcursor(wnd_client,20,19);
        wprintf(wnd_client,"%d of %u  ",clientpoint+1,nclient);
        show_visitnumber();
    }
}

```

```

void allfiles()
{
    WINDOW *wnd_size;
    long int i,totalHDspace,freeHDspace;
    allsize=0;
    allfilecount=0;
    for (i=0; i<nclient; i++) sizeclientfiles(i,0);
    allsize=nclient*220+allsize;
    allfilecount++;
    wnd_size = establish_window(33,10,9,43);
    showline("msg_ClientDirSiz");
    set_title(wnd_size,msg_line);
    set_colors(wnd_size,ALL,BLACK,YELLOW,BRIGHT);
    display_window(wnd_size);
    totalHDspace=get_disk_size('C');
    freeHDspace=get_disk_free('C');
    showline("msg_TotalSizes ");
    wprompt(wnd_size,4,1,msg_line);
    wprintf(wnd_size," %#9ld ",allsize);
    showline("msg_TotalFiles ");
    wprompt(wnd_size,12,2,msg_line);
    wprintf(wnd_size," %#9d",allfilecount);
    showline("msg_TotalHDspace");
    wprompt(wnd_size,1,4,msg_line);
    wprintf(wnd_size," %#9ld",totalHDspace);
    showline("msg_FreeHDspace ");
    wprompt(wnd_size,2,5,msg_line);
    wprintf(wnd_size," %#9ld",freeHDspace);
    get_char();
    delete_window(wnd_size);
}

```

```
newclient();
}
```

```
int spaces(char *c)
{
while (*c == ' ') c++;
return !*c;
}
```

```
int data_entry(WINDOW *wnd)
{
int exitcode, isvalid, done=FALSE, oldhelpkey=helpkey;
cursoron();
field_tally(wnd);
fld = FHEAD;
/* ---- collect data from keyboard into screen ---- */
while (fld != NULL && done == FALSE) {
    setmenuhelp(fld->fhwin, fld->flx, fld->fly);
    helpkey = (fld->fhelpt) ? 0 : oldhelpkey;
    wcursor(wnd, fld->fcol, fld->frow);
    if (fld->fprot == FALSE) {
        reverse_video(wnd);
        data_value(wnd, fld);
        wcursor(wnd, fld->fcol, fld->frow);
        exitcode = read_field(wnd, fld);
        isvalid = (exitcode != ESC && fld->fvalid) ?
                    (*(fld->fvalid))(fld->fbuff) : OK;
    }
    else {
        exitcode = FWD;
        isvalid = OK;
    }
    if (isvalid == OK) {
        normal_video(wnd);
        data_value(wnd, fld);
        switch (exitcode) {
            case F1:
                if (fld->fhelpt) {
                    /* passed edit */
                    (*(fld->fhelpt))(fld->fbuff);
                    data_value(wnd, fld);
                }
                break;
            case DN:
            case 'r':
            case 't':
            case FWD:
                fld = fld->fnxt;
        }
    }
}
}
```

```

                                if (fld == NULL) fld = FHEAD;
                                break;
        case UP:
        case BS:    fld = fld->fprv;
                    if (fld == NULL) fld = FTAIL;
                    break;
        default:    done = endstroke(exitcode);
                    break;
    }
}

helpkey = oldhelpkey;
return(exitcode);
}

void selectclient()
{
    int key;
    newclient();
    opmenu();
    while (key != ESC) {
        cursoroff();
        key = get_char();
        key = toupper(key);
        switch (key) {
            case PGDN:    clientpoint++;
                        if (clientpoint > nclient-1) clientpoint=0;
                        newclient();
                        break;
            case PGUP:    clientpoint--;
                        if (clientpoint < 0)
                            if (nclient > 0) clientpoint=nclient-1;
                            else clientpoint=0;
                        newclient();
                        break;
            case F2:    clientlist();
                        newclient();
                        setmenuhelp("client ",0,0);
                        break;
            case F3:    archivevisit(); break;
            case F4:    lastvisit(); break;
            case F5:    firstvisit(); break;
            case F6:    visitdates();
                        setmenuhelp("client ",0,0);
                        break;

```

```

        case 'E': if (nclient == clientpoint) break;
                    delete_window(wnd_infokeep);
                    data_entry(wnd_client);
                    writename(clientpoint,0);
                    opmenu();
                    break;
        case INS:   delete_window(wnd_infokeep);
                    zeroclient();
                    client.visits = 0;
                    show_visitnumber();
                    data_entry(wnd_client);
                    insertname();
                    setmenuhelp("client ",0,0);
                    newclient();
                    opmenu();
                    break;
        case DEL:   if (nclient < 0) break;
                    deletename();
                    newclient();
                    break;
        case 'N':   notepad();
                    setmenuhelp("client ",0,0);
                    break;
        case 'P':   printclientinfo();
                    break;
        case F9:    backuptoflop(); break;
        case F10:   restorefromflop();
                    newclient();
                    break;
        case F7:    sizeclientfiles(clientpoint,1); break;
        case F8:    allfiles(); break;
/*
        case '4':
        case '$': billinfo(); break;*/
        default: break;
    }
}
}

```

```

void showclient()
{
    clientcount();
    showinfo();
    ifclient='C';
    selectclient();
    cursoroff();
}

```



```

init_template(wnd_client);
/*xxxdelete_window(wnd_client);
delete_window(wnd_infokeep);*/
close_all();
}

```

```

/* FIND OPS */

```

```

void getsort(long int position)
{
int fhandle;
if ((fhandle=open("MASTER.SRT",O_RDONLY|O_BINARY)) == -1) {
    operationinfo("errorop",10,5,iskey);
    return;
}
lseek(fhandle,33*position,SEEK_CUR);
if ((read(fhandle,diskbuffer,33)) == -1) {
    operationinfo("errorfl",10,5,iskey);
    exit(1);
}
close(fhandle);
strncpy(name,diskbuffer,31);
}

```

```

void finditem()
{
char sameletter;
char notfound = 1;
int length,i,factor;
unsigned long int j;
unsigned long int limit;
char msk30[] = "_____";
clearfindname();
wnd_find = establish_window(3,12,3,32);
showline("msg_FindName ");
set_title(wnd_find,msg_line);
set_colors(wnd_find,ALL,GREEN,AQUA,BRIGHT);
set_colors(wnd_find,ACCENT,WHITE,BLACK,DIM);
display_window(wnd_find);
fld = establish_field(wnd_find,0,0,msk30,findname,'a');
cursoron();
ifclient='F';
data_entry(wnd_find);
cursoroff();
j=nitem/2;

```

```

limit=j/2;
length=strlen(findname);
for (i=0; i<length; i++) findname[i] = toupper(findname[i]);
strcpy(savefind,findname);
if (length == 0) {
    twobells();
    free(fld);
    delete_window(wnd_find);
    return;
}
factor=1;
while (notfound==1) {
    factor++;
    i=0;
    getsort(j);
    qfindnum=j;
    sameletter=1;
    while (sameletter) {
        while (findname[i] == name[i]) i++;
        if (i >= length) {
            sameletter=0;
            notfound=0;
            break;
        }
        sameletter=0;
        if (findname[i] < name[i]) j=j-limit;
        if (findname[i] > name[i]) j=j+limit;
        limit=limit/2;
        if (limit==0) limit=1;
        if (j > nitem) j=nitem-1;
        if (factor > 50) {
            sameletter=0;
            notfound=9;
            break;
        }
    }
}
if (notfound == 0) {
    itempoint=diskbuffer[31] + diskbuffer[32]*256;
    if (qfindnum > 0) {
        j=qfindnum-1;
        getsort(j);
        while (strncmp(name,findname,length) == 0) {
            if (j == 0) break;
            itempoint=diskbuffer[31] + diskbuffer[32]*256;

```

```

                j--;
                getsort(j);
            }
            if (j != qfindnum-1) qfindnum=j+1;
        }
    }
    else operationinfo("Ntfound",10,10,nokey);
    free(fld);
    delete_window(wnd_find);
    display20item();
    outitem();
}

```

```

void findq()
{
    int length;
    length=strlen(savefind);
    if (length == 0) {
        twobells();
        return;
    }
    qfindnum++;
    getsort(qfindnum);
    if (strncmp(name,savefind,length) == 0) {
        itempoint=diskbuffer[31] + diskbuffer[32]*256;
    }
    else {
        qfindnum--;
        beep();
    }
    display20item();
    outitem();
}

```

```

void findnum()
{
    int length,i;
    unsigned long int j;
    unsigned long int limit;
    char msk7[] = "_____";
    char findnum[8];
    for (i=0; i<8; i++) findnum[i] = 0;
    wnd_find = establish_window(3,12,3,13);
    showline("msg_FindNumber ");
    set_title(wnd_find,msg_line);
}

```

```

set_colors(wnd_find,ALL,GREEN,AQUA,BRIGHT);
set_colors(wnd_find,ACCENT,WHITE,BLACK,DIM);
display_window(wnd_find);
fld = establish_field(wnd_find,2,0,msk7,findnum,'a');
cursoron();
ifclient='F';
data_entry(wnd_find);
cursoroff();
length=strlen(findnum);
if (length == 0) {
    beep();
    free(fld);
    delete_window(wnd_find);
    return;
}
operationinfo("Searchg",10,10,nokey);
for (j=itempoint; j<nitem; j++) {
    getproduct(j);
    if (kbhit()!=0) break;
    if (strncmp(id,findnum,length) == 0) {
        itempoint=j;
        break;
    }
}
if (j>=nitem) {
    operationinfo("Ntfound",10,10,nokey);
    beep();
}
beep();
free(fld);
delete_window(wnd_find);
display20item();
outitem();
}

void jumpname()
{
char msk8[] = "_____";
WINDOW *wnd_jump;
wnd_jump = establish_window(2,22,3,9);
showline("msg_Jump      ");
set_title(wnd_jump,msg_line);
set_colors(wnd_jump,ALL,BLACK,AQUA,DIM);
set_colors(wnd_jump,ACCENT,WHITE,BLACK,DIM);
display_window(wnd_jump);

```

```

clearfindname();
fld = establish_field(wnd_jump,0,0,msk8,findname,'a');
cursoron();
ifclient='J';
data_entry(wnd_jump);
cursoroff();
free(fld);
delete_window(wnd_jump);
}

```

```

void enterpot()
{
char msk5[] = "_____";
int i;
WINDOW *wnd_entrpot;
wnd_entrpot = establish_window(30,0,3,10);
showline("msg_Enter   ");
set_title(wnd_entrpot,msg_line);
set_colors(wnd_entrpot,ALL,WHITE,BLACK,DIM);
display_window(wnd_entrpot);
for (i=0; i<12; i++) potname[i] = 0;
fld = establish_field(wnd_entrpot,0,0,msk5,potname,'A');
cursoron();
ifclient='J';
data_entry(wnd_entrpot);
cursoroff();
free(fld);
delete_window(wnd_entrpot);
}

```

```

/* AREA A-Z */

```

```

selectAZarea(void)
{
char save_screen[25*80*2];
int point;
int i,j,key;
int buffer[26];
char string[36];
gettext(1,1,80,25,save_screen);
for (i=0; i<26; i++) buffer[i]=0;
setgraphmode(g_mode);
setbkcolor(LIGHTBLUE);
setviewport(40,10,400,470,1);
DrawBorder();

```

```

setcolor(BROWN);
point=areapoint;
for (j='A'; j< '['; j++) {
    for (i=0; i<narea-1; i++) {
        if (j == area[i].select) {
            sprintf(string,"%c. %s",j,area[i].name);
            outtextxy(10,10+(j-65)*14,string);
            buffer[j-65]=i;
        }
    }
}
Bibox(2,130,412);
key = get_char();
key=toupper(key);
if (key > '@' && key < '[') {
    key = key-65;
    point = buffer[key];
}
backtext();
puttext(1,1,80,25,save_screen);
return point;
}

```

```

void showareas(char select,int aposition)
{
    char entrychar;
    int i,j,tempx,tempy,add;
    entrychar='A';
    i=first;
    for (j=0; j<46; j++) {
        if (i > narea-1) break;
        tempx=0;
        tempy=j;
        if (j > 22) {
            tempx=40;
            tempy=j-23;
        }
        /* gotoxy(1,1);
        printf("POSITION= %#3d i= %#3d first= %#3d",aposition,i,first);*/
        wcursor(wnd_prod,tempx,tempy);
        if (area[i].biblio == 'x') wprintf(wnd_prod," ");
        add = 0;
        if (select)
            if (aposition==i) reverse_video(wnd_prod);
        if (entrychar > 'Z') add = 6;
    }
}

```

```

        wputchar(wnd_prod,entrychar+add);
        wprintf(wnd_prod," %s",area[i].name);
        if (area[i].biblio == 'X') wprintf(wnd_prod," ");
        if (area[i].select != '/')
            wprintf(wnd_prod," %c",area[i].select);
        entrychar++;
        i++;
        normal_video(wnd_prod);
    }
    cursoroff();
}

selectarea()
{
    int i,key=0;
    int point;
    int areaposition=0;
    char select=0;
    setmenuhelp("AZarea ",12,5);
    wnd_prod = establish_window(0,0,25,80);
    set_colors(wnd_prod,ALL,6,BROWN,BRIGHT);
    set_colors(wnd_prod,ACCENT,WHITE,BLACK,DIM);
    display_window(wnd_prod);
    first=0;
    point=areapoint;
    showareas(0,0);
    while (key != ESC) {
        key = get_char();
        if (key > '@' && key < '{') {
            if (select) {
                key=toupper(key);
                if (key < '[') {
                    area[areaposition].select = key;
                    showareas(select,areaposition);
                    writeareafile();
                    beep();
                }
            }
            else {
                if (key > 'Z') key = key-6;
                if (key < narea+65) {
                    point=first+key-65;
                    break;
                }
            }
            else twobells();
        }
    }
}

```

```

    }
}
switch(key) {
    case PGUP: if (first-46 < 0) break;
               first=first-46;
               areaposition=first;
               clear_window(wnd_prod);
               showareas(select,areaposition);
               break;
    case PGDN: if (first + 46 > narea) break;
               first=first+46;
               areaposition=first;
               clear_window(wnd_prod);
               showareas(select,areaposition);
               break;
    case HOME: areaposition=first;
               showareas(select,areaposition);
               break;
    case END:  areaposition=first+45;
               if (areaposition > narea-1) areaposition=narea-1;
               showareas(select,areaposition);
               break;
    case CTRL_HOME: first=0;
                    clear_window(wnd_prod);
                    areaposition=first;
                    showareas(select,areaposition);
                    break;
    case CTRL_END: first=narea-46;
                   if (first<0) first=0;
                   areaposition=first;
                   clear_window(wnd_prod);
                   showareas(select,areaposition);
                   break;
    case INS: if(select) select=0;
              else select++;
              showareas(select,areaposition);
              break;
    case DEL: area[areaposition].select = '/';
              clear_window(wnd_prod);
              showareas(select,areaposition);
              writeareafile();
              beep();
              break;
    case UP:  if (areaposition != first) areaposition--;
              showareas(select,areaposition);

```



```

        break;
    case DN: if (areaposition != first+45) areaposition++;
             if (areaposition > narea-1) areaposition--;
             showareas(select,areaposition);
             break;
    case BS:
    case FWD: areaposition = areaposition + 23;
             if (areaposition > first+45) areaposition = areaposition-46;
             if (areaposition > narea-1) areaposition=areaposition-23;
             showareas(select,areaposition);
             break;
    default: break;
}
}
delete_window(wnd_prod);
return (point);
}

```

```
/* INPUT and SOUND */  
/* Copyright 1992-8 James Hoyt Clark */
```

```
#include <time.h>  
#include <stdio.h>  
#include <dos.h>  
#include <stdlib.h>  
#include <math.h>  
#include <conio.h>  
#include <ctype.h>  
#include <io.h>  
#include <fcntl.h>  
#include <graphics.h>  
#include "keys.h"  
#include "twindow.h"  
#include "pcl4c.h"
```

```
#define INDEX 100
```

```
int nSerFlag=1;  
int nCount4 = 0;  
int nCount5 = 0;  
int nCountff = 0;  
int nCount1 = 0;  
int nTimes=0;  
int nBuffer[INDEX];  
unsigned char waveamplitude[256];  
unsigned char plotamp[256];  
unsigned char voltage;  
unsigned char gain;  
int wavepoint;  
int timer0;  
int timer1;  
int timer2;  
char freqname[7];  
float freqvalue;  
int freqpoint;  
int freqcount;  
int freqsave;  
unsigned char portcstat;  
WINDOW *wnd_panel;  
int TICS = 1;  
const DELAY = 10;  
int CARD;  
int PORTABLE;
```

```
int nBaud = Baud9600;
int nDTime = 8;
int nPortDelay;
int nGetBuffer = 4;
extern char far TxBuffer[16384];
extern char far RxBuffer[128];
extern int defaultvalues[100];
```

```
/* DCM Sound */
```

```
char multi_am;
char multi_vib;
char multi_egtyp;
char multi_ksr;
char multi;
```

```
char tone_ksl;
char tone_level;
char tone_dc;
char tone_dm;
char tone_fb;
```

```
char attack;
char decay;
```

```
char suslevel;
char relrate;
```

```
char rhy_mel;
char drum_BD;
char drum_SD;
char drum_TOM;
char drum_TCY;
char drum_HH;
```

```
unsigned char pitch;
```

```
char susON_OFF;
char keyON_OFF;
char octave;
char f_num8;
```

```
char instrument;
char volume;
```

```
char percussion;
```

```
char bd_vol;  
char hh_vol;  
char sd_vol;  
char tom_vol;  
char tcy_vol;
```

```
extern unsigned char input;  
extern int reading;  
extern char pointfindflg;  
extern int bkcolor;  
extern float graphscale;  
extern char ifgraphic;  
extern int SerPort;  
extern int nCounter;
```

```
void reset193(void)  
{  
    outportb(PPIPORTC,portcstat | 0x04);    /* Assert 193 RESET */  
    delay(400);  
    outportb(PPIPORTC,portcstat & 0xFB);    /* No interrupt */  
}
```

```
void memDCMCard(void)  
{  
    int i;  
    int board;  
    for (i=0; i<100; i++)  
    {  
        board = inportb(ADPORTE);  
        printf("\nCHAR: %d\n",board);  
        if (board == 0)        break;  
    }  
    if (board != 0) exit(0);  
}
```

```
void setPCB(void)  
{  
    portcstat=0x08; /* Sound off */  
    disable();  
    outportb(PPICTLREG,0x92); /* A=IN, B=OUT, C=OUT */  
    outportb(PPIPORTC,portcstat);    /* No sound */  
    outportb(PPIPORTB,0);  
    outportb(ADPORTEF,0x44); /* Prog Chan0 8 bits + sign*/  
}
```

```

outportb(CTRLWRD,0x36); /* Prog counters to Mode 3 */
outportb(CTRLWRD,0x76);
outportb(CTRLWRD,0xB6);
outportb(SETGAIN,gain); /* Set gain */
outportb(SETVOLT,voltage); /* Set voltage */
reset193();
memDCMCard();
}

```

```

/* Audio */

```

```

char amp;
char savamp;
char chan=0; /* Audio Channel */

```

```

/* SOUND */

```

```

void outsound(char address, char data)
{
outportb(SOUNDADDR,address);
delay (2);
outportb(SOUNDDATA,data);
delay (2);
}

```

```

void onvolume()
{
unsigned char value;
amp=savamp;
if (CARD == KEYCARD)
{
outsound(INST_VOL, (instrument << 4) | 15-(amp/2));
portcstat = portcstat | 0x08;
outportb(PPIPORTC,portcstat);
}
}

```

```

void offvolume()
{
unsigned char value;
if (CARD == KEYCARD)
{
outsound(V_BLOCK, 0x05);
portcstat = portcstat & 0xF3;
outportb(PPIPORTC,portcstat);
}
}

```

```

    }
    if (PORTABLE == KEYPORT)
        nosound();
    }

    void setpitch(char pitch,char octv,char inst)
    {
        outsound(V_FNUM, pitch);
        outsound(INST_VOL, (inst << 4) | 15-(amp/2));
        outsound(V_BLOCK, susON_OFF | keyON_OFF | (octv << 1) | f_num8);
    }

    void xsetpitch(char pitch)
    {
        setpitch(pitch,4,instrument);
    }

    /* DCM INPUT */

    void GetCard(void) // DCM Card
    {
        char status;
        int i,j;
        char string[60];
        unsigned char valuF=0;
        unsigned char valuE=0;
        unsigned char zeroscale;
        unsigned char oldinput;
        unsigned long total=0;
        int count=10;
        int counter=0;
        unsigned char time=0;
        unsigned char buffer[10];
        unsigned char input0;
        unsigned char input1;
        unsigned char input2;
        unsigned char input3;
        unsigned char input4;

        nSerFlag = 1;
        zeroscale=255-voltage;
        for (i=0; i<count; i++) {
            outportb(SETGAIN,gain); /* Set gain */
            outportb(ADPORTF,0x44); /* Prog Chan0 8 bits + sign*/
            while (time < 10) {

```

```

        if (!(status = inportb(ADSTAT) & 0x01))    break; /* finished */
        delay(4);
        time++;
    }
    if (status == 0) {
        valuE = inportb(ADPORTE);
        valuF = inportb(ADPORTF);
        if (valuE == 255)
            if (wavepoint) valuF = 255-valuF;
            else break;
        input = valuF;
        total=total+input;
        buffer[counter] = valuF;
        counter++;
    }
}
if ((voltage) && (counter > 1)) {
    i=(total-buffer[0])/(counter-1);
    if (i < 3) input=255;
    else input = 258 - i; /* reverse order */
    input0=input;
    if (input < zeroscale) input = 0;
    else input = input-zeroscale; /* shift 0 */
    input1=input;
    input = input*(255.0/voltage); /* scale 0-255 */
    input2=input;
    input3=input;
    reading=input/2.55; /* scale 0-100 */
    input4=reading;
    oldinput=input;
}
else {
    input = oldinput;
    reading = input/2.55;
}
if (oldinput != input) {
    if (ifgraphic) {
        setcolor(BLUE);
        setviewport(0,0,639,479,1);
        rectangle(4,39,481,101);
        setviewport(5,40,480,100,1);
        clearviewport();
        settextstyle(SMALL_FONT,HORIZ_DIR,4);
        setcolor(WHITE);
        sprintf(string,"Z=%#4d R=%#4d Total= %#3d<",

```

```

        zeroscale,reading,total);
    outtextxy(10,0,string);
    sprintf(string,"I=%#4d V=%#4d E=%#4d F=%#4d C=%#2d<",
        input,voltage,valuE,valuF,counter);
    outtextxy(10,15,string);
    for (j=0; j<counter; j++) {
        sprintf(string,"%#2d %#4d",j,buffer[j]);
        outtextxy(40*j+10,30,string);
    }
    sprintf(string,"0=%#4d 1=%#4d 2= %#4d 3= %#4d 4= %#4d <",
        input0,input1,input2,input3,input4);
    outtextxy(10,45,string);
    setviewport(0,0,639,479,1);
}
else {
    gotoxy(10,10);
    printf("\nIN= %#4d V= %#4d Z=%#4d E= %#4d F= %#4d C=%#2d Total=%#6d
<",
        input,voltage,zeroscale,valuE,valuF,counter,total);
    gotoxy(1,1);
    for (j=0; j<counter; j++) {
        printf("\n%#2d %#3d",j, buffer[j]);
    }
    gotoxy(1,15);
    printf("0=%#4d 1=%#4d 2= %#4d 3= %#4d 4= %#4d <",
        input0,input1,input2,input3,input4);
}
}
if (reading > 3) setpitch(input,4,instrument);
if (reading > 35) onvolume();
else if (pointfindflg == 0) {
    savamp=amp;
    offvolume();
}
}

/* DCM SHIFT */

void shiftcode(char number)
{
    int i;
    outportb(SHIFT,number);
    for (i=0; i<8; i++) inportb(CLKSFT1);
}

```



```

void shiftpot(char number)
{
    int i;
    outportb(SHIFT,number);
    for (i=0; i<8; i++) inportb(CLKSFT2);
}

```

```

/* DCM SET TIMER MODES */

```

```

void getfrequency(int position)
{
    char diskbuffer[23];
    char string[7];
    int fhandle,i,j;
    if ((fhandle=open("FREQLIST.DAT",O_RDONLY|O_BINARY)) == -1) {
        printf("open failed");
        exit(1);
    }
    lseek(fhandle,23*position,SEEK_CUR);
    if ((read(fhandle,diskbuffer,23)) == -1) {
        perror("read error");
        exit(1);
    }
    j=0;
    for (i=0; i<4; i++) {
        string[j]=diskbuffer[i];
        j++;
    }
    string[4]=0;
    timer0=atoi(string);
    j=0;
    for (i=5; i<9; i++) {
        string[j]=diskbuffer[i];
        j++;
    }
    string[4]=0;
    timer1=atoi(string);
    j=0;
    for (i=10; i<14; i++) {
        string[j]=diskbuffer[i];
        j++;
    }
    string[4]=0;
    timer2=atoi(string);
    j=0;
}

```

```

for (i=15; i<21; i++) {
    string[j]=diskbuffer[i];
    j++;
}
string[6]=0;
freqvalue=atof(string);
strcpy(freqname,string);
close(fhandle);
}

```

```

void changetimer(unsigned PLACE, int timer)
{
    unsigned char numhigh;
    unsigned char numlow;
    numlow=timer;
    numhigh=timer/0x100;
    outportb(PLACE,numlow);
    outportb(PLACE,numhigh);
}

```

```

void tellfrequency(void)
{
    char string[10];
    int i,j,xbar;
    float range;
    getfrequency(freqpoint);
    changetimer(TIMER0,timer0);
    changetimer(TIMER1,timer1);
    changetimer(TIMER2,timer2);
    setcolor(WHITE);
    rectangle(483,351,562,380);
    setviewport(484,352,561,379,1);
    clearviewport();
    settextstyle(TRIPLEX_FONT,HORIZ_DIR,2);
    outtextxy(4,0,freqname);
    settextstyle(SMALL_FONT,HORIZ_DIR,USER_CHAR_SIZE);

```

```

/*setviewport(0,0,639,479,1);
rectangle(0,0,40,32);
setviewport(1,1,39,31,1);
clearviewport();
sprintf(string,"%#4d",timer0);
outtextxy(1,0,string);
sprintf(string,"%#4d",timer1);
outtextxy(1,10,string);

```

```

sprintf(string, "%#4d", timer2);
outtextxy(1, 20, string); */

if (freqvalue < 10000)
    if (freqvalue < 1000)
        if (freqvalue < 100)
            if (freqvalue < 10)
                if (freqvalue < 1) range = 0.1;
                else range = 1;
            else range = 10;
        else range = 100;
    else range = 1000;
else range = 10000;
j=range;
setviewport(0, 0, 639, 479, 1);
setfillstyle(SOLID_FILL, BLUE);
bar(2, 353, 480, 363);
for (i=0; i<11; i++) {
    if (range < 1) sprintf(string, "%#1.1f", i*range);
    else sprintf(string, "%d", i*j);
    outtextxy(45*i+2, 353, string);
    line(45*i+4, 369, 45*i+4, 365);
}
xbar = freqvalue * 45.0/range;
setfillstyle(SOLID_FILL, RED);
bar(5, 370, xbar+5, 378);
setfillstyle(SOLID_FILL, WHITE);
bar(xbar+5, 370, 454, 378);
}

```

```

void topfrequency(void)
{
    freqpoint=freqcount-1;
    tellfrequency();
}

```

```

void lowfrequency(void)
{
    freqpoint=1;
    tellfrequency();
}

```

```

void zerofrequency(void)
{
    freqpoint=0;
}

```

```
tellfrequency();  
}
```

```
void afrequency(void)  
{  
    freqpoint=25;  
    tellfrequency();  
}
```

```
void increasefreq()  
{  
    if (!wavepoint) return;  
    if (freqpoint == freqcount-1) return;  
    freqpoint++;  
    tellfrequency();  
}
```

```
void decreasefreq()  
{  
    if (!wavepoint) return;  
    if (freqpoint == 0) return;  
    freqpoint--;  
    tellfrequency();  
}
```

```
/* DCM WAVE SHAPES */
```

```
void transmitwave(void)  
{  
    outportb(SETVOLT,voltage);  
    portcstat = portcstat & 0xFE;  
    outportb(PPIPORTC,portcstat);  
}
```

```
void stopwave(void)  
{  
    outportb(SETVOLT,0);  
    portcstat = portcstat | 0x01;  
    outportb(PPIPORTC,portcstat);  
}
```

```
void writewavetoram(void)  
{  
    int i;  
    unsigned int DATA;
```

```

stopwave();
for (i=0; i<256; i++) {
    DATA = i*0x100;
    DATA = DATA + waveamplitude[i];
    output(RAMPORT,DATA);
}
transmitwave();
}

```

```

void dcline(void)
{
int i;
wavepoint=0;
for (i=0; i<256; i++) {
    waveamplitude[i] = 255;
    plotamp[i] = voltage;
}
writewavetoram();
}

```

```

void sinewave(void)
{
int i;
double angle;
wavepoint=1;
for (i=0; i<256; i++) {
    angle=i/2.40/RAD_DEG;
    waveamplitude[i] = sin(angle) * i;
    plotamp[i] = waveamplitude[i]*voltage/255;
}
writewavetoram();
}

```

```

void squarewave(void)
{
int i;
wavepoint=2;
for (i=0; i<128; i++) {
    waveamplitude[i] = 0;
    plotamp[i] = 0;
}
for (i=128; i<256; i++) {
    waveamplitude[i] = 255;
    plotamp[i] = voltage;
}
}

```

```
writewavetoram();  
}
```

```
void trianglewave(void)  
{  
    int i;  
    wavepoint=3;  
    for (i=0; i<256; i++) {  
        waveamplitude[i] = i;  
        plotamp[i] = i * voltage/255;  
    }  
    writewavetoram();  
}
```

```
void getselectwave(void)  
{  
    switch (wavepoint) {  
        case 0: dcline(); break;  
        case 1: sinewave(); break;  
        case 2: squarewave(); break;  
        case 3: trianglewave(); break;  
    }  
}
```

```
void whichwave(void)  
{  
    char string[10];  
    float number;  
    outportb(SETVOLT,voltage);  
    number=voltage;  
    setcolor(WHITE);  
    rectangle(483,305,535,334); /*Voltage Box*/  
    setviewport(484,306,534,333,1);  
    clearviewport();  
    settextstyle(TRIPLEX_FONT,HORIZ_DIR,2);  
    sprintf(string,"%#.1.2f",number*.0196);  
    outtextxy(3,0,string);  
    setviewport(0,0,639,479,1);  
    setfillstyle(SOLID_FILL,RED);  
    bar(5,320,number*1.7647+5,328);  
    setfillstyle(SOLID_FILL,WHITE);  
    bar(number*1.7647+5,320,454,328);  
    getselectwave();  
    plotwave();  
}
```

```

void selectwave(void)
{
    wavepoint++;
    if (wavepoint > 3) wavepoint=0;
    if (wavepoint == 1) {
        freqpoint=freqsave;
        tellfrequency();
    }
    if (wavepoint == 0) {
        freqsave=freqpoint;
        zerofrequency();
    }
    whichwave();
}

```

```

void tellgain(void)
{
    char string[2];
    outportb(SETGAIN,gain);
    setcolor(WHITE);
    rectangle(280,416,300,443);
    setviewport(281,417,299,442,1);
    clearviewport();
    settextstyle(TRIPLEX_FONT,HORIZ_DIR,2);
    sprintf(string,"%d",gain);
    outtextxy(4,0,string);
    setviewport(0,0,639,479,1);
}

```

```

void tellinstrument(void)
{
    char string[3];
    setcolor(WHITE);
    rectangle(350,416,383,443);
    setviewport(351,417,382,442,1);
    clearviewport();
    settextstyle(TRIPLEX_FONT,HORIZ_DIR,2);
    sprintf(string,"%2d",instrument);
    outtextxy(4,0,string);
    setviewport(0,0,639,479,1);
}

```

```

void setout()
{
    set_colors(wnd_panel,ALL,RED,BLUE,BRIGHT);
}

```

```
}
```

```
void setback()
```

```
{  
set_colors(wnd_panel,ALL,MAGENTA,GREEN,BRIGHT);  
}
```

```
void soundpanel(void)
```

```
{  
int key=0;  
int i;  
wnd_panel = establish_window(0,0,25,80);  
set_colors(wnd_panel,ALL,MAGENTA,GREEN,BRIGHT);  
set_colors(wnd_panel,ACCENT,BLUE,WHITE,BRIGHT);  
display_window(wnd_panel);  
while (key != ESC) {  
    wcursor(wnd_panel,0,0);  
    wprintf(wnd_panel,"6=AM 7=VIB 8=EG-TYP 9=KSR 0=Inc MULTI )=Dec  
MULTI");  
    wprintf(wnd_panel,"\n %#2d  %#2d  %#2d  %#2d  %#2d",  
            (multi_am >> 7),(multi_vib >> 6),(multi_egtyp >> 5),  
            (multi_ksr >> 4),multi);  
    wprintf(wnd_panel,"\nLevel key scale (0-3) (k)");  
    wprintf(wnd_panel," = %#2d",tone_ksl);  
    wprintf(wnd_panel,"\nTone Level (0-64) Inc (I) Dec (L)");  
    wprintf(wnd_panel," %#2d",tone_level);  
    wprintf(wnd_panel,"\nh= Modulated wave half= %#2d j= Carrier wave half=  
%#2d",tone_dc,tone_dm);  
    wprintf(wnd_panel,"\nb= Increase Feedback (0-3)");  
    wprintf(wnd_panel," = %#2d",tone_fb);  
  
    wprintf(wnd_panel,"\na= Increase Attack Rate  A= Decrease Attack Rate");  
    wprintf(wnd_panel," = %#2d",attack);  
    wprintf(wnd_panel,"\nd= Increase Decay Rate  D= Decrease Decay Rate");  
    wprintf(wnd_panel," = %#2d",decay);  
  
    wprintf(wnd_panel,"\nu= Increase Sustain Level  U= Decrease Sustain Level");  
    wprintf(wnd_panel," = %#2d",suslevel);  
    wprintf(wnd_panel,"\ne= Increase Release Rate  E= Decrease Decay Rate");  
    wprintf(wnd_panel," = %#2d",relrate);  
  
    wprintf(wnd_panel,"\nm = Rhythm/Melody");  
    wprintf(wnd_panel," = %#2d",(rhy_mel >> 5));  
  
    wprintf(wnd_panel,"\np = Increase Pitch      P= Decrease Pitch");
```



```

wprintf(wnd_panel, " Pitch = %u ",pitch);

wprintf(wnd_panel, "\ns = Sustain ON/OFF = %#2d   w= Key ON/OFF = %#2d",
(susON_OFF >> 5),(keyON_OFF >> 4));
wprintf(wnd_panel, "\nOctave Inc (o) Dec (O) %#3d  Octave (f) %#2d",
octave,f_num8);

wprintf(wnd_panel, "\n\nInstrument (i) = %#3d",instrument);
wprintf(wnd_panel, "   Inc (+) Dec (-) Volume = %#3d",volume);

wprintf(wnd_panel, "\n\n***** PERCUSSION *****");
wprintf(wnd_panel, "\n(1) %#2d Bass Drum  Inc (z) Dec (Z) ",drum_BD >> 4);
setout();
wprintf(wnd_panel, "Vol = %#2d",bd_vol);
setback();
wprintf(wnd_panel, "\n(2) %#2d Snare.Drum Inc (c) Dec (C) ",drum_SD >> 3);
setout();
wprintf(wnd_panel, "Vol = %#2d",sd_vol);
setback();
wprintf(wnd_panel, "\n(3) %#2d Tom-tom   Inc (v) Dec (V) ",drum_TOM >> 2);
setout();
wprintf(wnd_panel, "Vol = %#2d",tom_vol);
setback();
wprintf(wnd_panel, "\n(4) %#2d Top Cymbal Inc (g) Dec (G) ",drum_TCY >> 1);
setout();
wprintf(wnd_panel, "Vol = %#2d",tcy_vol);
setback();
wprintf(wnd_panel, "\n(5) %#2d High Hat   Inc (x) Dec (X) ",drum_HH);
setout();
wprintf(wnd_panel, "Vol = %#2d",hh_vol);
setback();
key=getch();
switch(key) {
    case '6': if (multi_am==0) multi_am=0x80;
              else multi_am=0;
              break;
    case '7': if (multi_vib==0) multi_vib=0x40;
              else multi_vib=0;
              break;
    case '8': if (multi_egtyp==0) multi_egtyp=0x20;
              else multi_egtyp=0;
              break;
    case '9': if (multi_ksr==0) multi_ksr=0x10;
              else multi_ksr=0;
              break;
}

```

```

        case '0': multi++;
if (multi > 15) multi = 0;
        break;
        case ')': multi--;
if (multi < 0) multi = 15;
        break;

case 'k': tone_ksl++;
        if (tone_ksl > 3) tone_ksl = 0;
        break;
        case 'l': tone_level++;
        if (tone_level > 63) tone_level = 0;
        break;
case 'L': tone_level--;
        if (tone_level < 0) tone_level = 63;
        break;
        case 'h': if (tone_dc==0) tone_dc=0x10;
else tone_dc=0;
        break;
        case 'j': if (tone_dm==0) tone_dm=0x08;
        else tone_dm=0;
        break;
        case 'b': tone_fb++;
        if (tone_fb > 3) tone_fb = 0;
        break;

        case 'a': attack++;
if (attack > 15) attack = 0;
        break;
        case 'A': attack--;
        if (attack < 0) attack = 15;
        break;
        case 'd': decay++;
        if (decay > 15) decay = 0;
        break;
        case 'D': decay--;
        if (decay < 0) decay = 15;
        break;

        case 'u': suslevel++;
        if (suslevel > 15) suslevel = 0;
break;
        case 'U': suslevel--;
        if (suslevel < 0) suslevel = 15;
break;

```

```

        case 'e': relrate++;
            if (relrate > 15) relrate = 0;
break;
        case 'E': relrate--;
            if (relrate < 0) relrate = 15;
            break;

        case 'm': if (rhy_mel==0) rhy_mel=0x20;
else rhy_mel=0;
break;
        case 'l': if (drum_BD==0) drum_BD=0x10;
            else drum_BD=0;
            break;
        case '2': if (drum_SD==0) drum_SD=0x08;
            else drum_SD=0;
            break;
case '3': if (drum_TOM==0) drum_TOM=0x04;
            else drum_TOM=0;
            break;
case '4': if (drum_TCY==0) drum_TCY=0x02;
            else drum_TCY=0;
            break;
        case '5': if (drum_HH==0) drum_HH=0x01;
            else drum_HH=0;
            break;

        case 'p': pitch++;
            break;
case 'P': pitch--;
            break;

        case 's': if (susON_OFF==0) susON_OFF=0x20;
else susON_OFF=0;
            break;
        case 'w': if (keyON_OFF==0) keyON_OFF=0x10;
            else keyON_OFF=0;
            break;
        case 'o': octave++;
            if (octave > 7) octave = 0;
break;
        case 'O': octave--;
            if (octave < 0) octave = 7;
break;
        case 'f': if (f_num8) f_num8=0;
            else f_num8=1;

```

```

        break;

        case 'i': instrument++;
if (instrument > 15) instrument = 0;
break;

        case '+': volume++;
        if (volume > 15) volume = 0;
        break;
        case '-': volume--;
        if (volume < 0) volume = 15;
        break;

        case 'z': bd_vol++;
        if (bd_vol > 15) bd_vol = 0;
break;

        case 'Z': bd_vol--;
        if (bd_vol < 0) bd_vol = 15;
break;

        case 'x': hh_vol++;
        if (hh_vol > 15) hh_vol = 0;
        break;
        case 'X': hh_vol--;
        if (hh_vol < 0) hh_vol = 15;
break;

        case 'c': sd_vol++;
        if (sd_vol > 15) sd_vol = 0;
        break;
case 'C': sd_vol--;
        if (sd_vol < 0) sd_vol = 15;
        break;
        case 'v': tom_vol++;
        if (tom_vol > 15) tom_vol = 0;
        break;
        case 'V': tom_vol--;
        if (tom_vol < 0) tom_vol = 15;
        break;
        case 'g': tcy_vol++;
        if (tcy_vol > 15) tcy_vol = 0;
        break;
        case 'G': tcy_vol--;
        if (tcy_vol < 0) tcy_vol = 15;
        break;
        case '*': onvolume(); /*outportb(PPIPORTC,portcstat & 0xF7);*/
        outsound(0x0e, 0);
        outsound(0, 0x17);

```

```

    outsound(1, 0x62);
    outsound(2, 0x5b);
    outsound(3, 0x06);
    outsound(4, 0xf1);
    outsound(5, 0xe0);
    outsound(6, 0xf2);
    outsound(7, 0xf2);
    outsound(0x10, 0xe5);
    outsound(0x20, 0x06);
    outsound(0x30, 0);
    outsound(0x20, 0x16);
    delay(2000);
    outsound(0x20, 0x06);
    i = 0;
    for (i = 0; i >= 0; ) {
        outsound(0x0e, 0);
        outsound(0, 0x21);
        outsound(1, 0x21);
        outsound(2, 0x3f);
        outsound(3, 0);
        outsound(4, 0xf0);
        outsound(5, 0xf0);
        outsound(6, 0x0f);
        outsound(7, 0x0f);
        outsound(0x30, i << 4);
        outsound(0x10, 0x20);
        outsound(0x20, 0x05);
        outsound(0x20, 0x15);
        while (!kbhit());
        i = getch();
        if (i >= '0' && i <= '9')
            i -= '0';
        else if (i >= 'a' && i <= 'f')
            i -= 'a' - 0x0a;
        else
            i = -1;
    }
    outsound(0x20, 0x05);
    offvolume(); /*outportb(PPIPORTC,portcstat | 0x08);*/
    break;
default: break;
}
onvolume();
outsound(V_MULTI, multi_am | multi_vib | multi_egtyp | multi_ksr | multi);
outsound(V_LEVEL2, (tone_ksl << 6) | tone_level);

```

```

    outsound(V_LEVEL3, (tone_ksl << 6) | tone_dc | tone_dm | tone_fb);
    outsound(V_ATDERATE, (attack << 4) | decay);
    outsound(V_SL_RR, (suslevel << 4) | relrate);
    percussion = drum_BD | drum_SD | drum_TOM | drum_TCY | drum_HH;
    outsound(RHY_CTL, rhy_mel | percussion);
    outsound(V_FNUM, pitch);
    outsound(V_BLOCK, susON_OFF | keyON_OFF | (octave << 1) | f_num8);
    outsound(INST_VOL, (instrument << 4) | volume);
    outsound(RHYTHM_VOL1, bd_vol);
    outsound(RHYTHM_VOL2, (hh_vol << 4) | sd_vol);
    outsound(RHYTHM_VOL3, (tom_vol << 4) | tcy_vol);
}
offvolume();
delete_window(wnd_panel);
}

```

```

void setscale(void)
{
    graphscale=1.575;// for DCM Card
}

```

/* SERIAL STUFF */

```

int GetPortable(void) // DCM-M
{
    int total=0;
    int i;
    int nInput;
    int nMinInput = 100;
    int nMaxInput = 0;

    for (i=0; i<INDEX; i++) nBuffer[i] = 0;
    for (i=0; i<INDEX; i++) // average some bytes
    {
        nInput = SioGetc(SerPort, TICS); // get byte
        if (!nInput) // filter zero
        {
            if (i) nInput = total/(i+1); // spurious if index not 0
            else nSerFlag = 1;
        }
        if (nInput == FREEZE) // filter FREEZE(-1)
        {
            UnFreeze();
            nInput = total/(i+1);
        }
    }
}

```

```

        if (nInput == 0xff)    // filter 0xff
        {
            UnFreeze();
            nInput = total/(i+1);
        }

        if (nInput)
        {
            nBuffer[i] = nInput;
            if (nMinInput > nInput) nMinInput = nInput;
            if (nMaxInput < nInput) nMaxInput = nInput;
        }

        if (nInput == RIGHT || nInput == LEFT) // if footpedal
        {
            nosound(); // turn off sound
            input = 0; // set graph line to 0
            nSerFlag = 0; // flag for footpedal
            return nInput; // return non reading
        }
        total = nInput + total;
    }
    if (nMinInput < 7 || nMaxInput+nMinInput < 15 ||
        nMaxInput/nMinInput > 2 || nBuffer[0]-nBuffer[99] > 5)
        nSerFlag = 0;
    else nSerFlag = 1;

/*
    if (nInput)
    {
        cursor(45,15);
        printf("X=%#3d N=%#3d S=%#2d p=%#3d n=%#4d d=%#3d",
            nMaxInput,nMinInput,nSerFlag,
            nMaxInput+nMinInput,nMaxInput-nMinInput,nMaxInput/nMinInput);
    }*/

    if (!nSerFlag)
    {
        nosound();
        total = 0;
    }

    nInput = (total/i);
    input = nInput*2.55;
    if (nInput > LOWREAD && nSerFlag && pointfindflg)
        sound(nInput*10);

```

```

        else if (nInput > 35) sound(nInput*10);
        if (nInput < LOWREAD) nosound();
        cursor(0,0);
//      if (nInput)
//          for (i=0; i<INDEX; i++)
//              printf("%#2d.%#3d ",i,nBuffer[i]);
//      SioRxFlush(SerPort);
        return nInput;
    }

```

```

void DoOFFTest()
{
    SioError(SioPutc(SerPort,SWITCHOFFAA));
    delay(nDTime*2);
}

```

```

void DoOFF()
{
    //delay(nPortDelay);
    SioPutc(SerPort,SWITCHOFFAA);
    delay(nPortDelay*6);
}

```

```

void DoONTest(void)
{
    int i;
        SioError(SioPutc(SerPort,SWITCHONCC));
        SioError(SioPutc(SerPort,SWITCHON55));
        SioError(SioPutc(SerPort,SWITCHON99));
        delay(nDTime*2);
}

```

```

void DoON(void)
{
    int i;
        SioBaud(SerPort,Baud9600);
        delay(nPortDelay*6);
        SioPutc(SerPort,SWITCHONCC);
        SioPutc(SerPort,SWITCHON55);
        SioPutc(SerPort,SWITCHON99);
//      delay(nPortDelay);
}

```

```

void getinput(void)
{

```



```

    if (CARD==KEYCARD)
    {
        GetCard();
        return;
    }
    if (PORTABLE==KEYPORT)
    {
        reading = GetPortable();
        if (reading == FREEZE)
        {
            delay(10);
            reading = GetPortable();
            if (reading == FREEZE)
            {
                UnFreeze();
                nTimes++;
                beep();
                beep();
                printf("T:%#4d",nTimes);
                reading = 0;
            }
        }
    }
}

```

```

void ClearDCM(void)
{
    // SioPutc(SerPort,0);
    // SioPutc(SerPort,0);
    SioDone(SerPort); // done with serial port
}

```

```

void DisplayReadBaud(int nRead)
{
    cursor(5,3);
    printf("Reading:%#4d",nRead);
    //if (nBaud == 5) printf("9600 ");
    //if (nBaud == 9) printf("115200");
}

```

```

void CheckFor(void)
{
    int nPut = 0x00;
    int nRead = -1;
    while (nRead == -1)

```

```

    {
        SioPutc(SerPort,nPut);
        nRead = SioGetc(SerPort,TICS);
        printf("\nRead: %#4d Put: %x",nRead, nPut);
        if (nPut == 0xff) break;
        nPut++;
    }
get_char();
}

void SerialSetup(void)
{
//    nGetBuffer = Size128;
    SioRxBuf(SerPort, RxBuffer, nGetBuffer);
    SioTxBuf(SerPort, TxBuffer, Size16K);
    SioParms(SerPort, NoParity, OneStopBit, WordLength8);
    SioReset(SerPort, Baud9600);
    SioDTR(SerPort, 'S');
    SioDelay(DELAY);

    SioDTR(SerPort, 'C'); // clear DTR
    SioDelay(DELAY);

/* SioError (SioRxBuf(SerPort, Buffer, Size128));
   SioError (SioParms(SerPort, NoParity, OneStopBit, WordLength8));
   SioError (SioReset(SerPort, Baud9600));
   SioError (SioDTR(SerPort, 'S'));
   SioError (SioDelay(DELAY));

   SioError (SioDTR(SerPort, 'C')); // clear DTR
   SioError (SioDelay(DELAY));*/

    SioPutc(SerPort,0xF0); // set to start
}

void SetupDCM(void)
{
    int i,j;
    int nRead;
    int nPut;
    int nKey;
    int nLookFor = 0;
    char cNum;
    //int nLow;
    //int nHigh;

```

```

int nHighCount;
int nCount3, nCount4, nCount5, nCount6;

    cursoroff();
    clrscr();
    printf("This software and hardware are designed for use as an ohmmeter");
    printf("\nand a signal generator. Their function is to be used by a");
    printf("\nCEDS operator for measuring energy balance or imbalance with");
    printf("\nthe ohmmeter and linking signals from the signal generator");
    printf("\nto a balance or imbalance measured with the ohmmeter.");
    printf("\n\nThe manufacturer makes no medical claims.");
    delay(5000);
    clrscr();
    printf("\nOn front of DCMR: Power light (green) ON.");
    printf("\n          Reset/Prgm light (red) ON.");
    printf("\n\nIf no red Reset light. Press red Reset button on back of DCMR.");
    delay(3000);

while (1)
{
    SerialSetup();
    nRead = SioGetc(SerPort, TICS);
    if (nRead != 0xF8)
    {
        SerPort++;
        SioDone();
        if (SerPort > 4) SerPort = 0;
    }
    if (nRead == 0xF8) break;
    if (SerPort == defaultvalues[7]) break;
}
if (nRead != 0xF8)
{
    printf("\nNo DCM Receiver detected. Code: %x", reading);
    printf("\nPlease check DCMR serial cable and power cable.");
    printf("\nIf Reset light is off, press Reset button on back of DCMR.");
    printf("\nThen start program again.");
    printf("\n\nC - DCM Card (Half size)");
    printf("\nP - DCM Portable");
    printf("\nS - Port");
    putchar(BELL);
    delay(3000);
    boogieout();
}
putchar(BELL);

```

```

printf("\nDCM Receiver detected at COM%d ",SerPort+1);
delay(2000);
printf("\n\nPort status: Clearing (0 to 30 seconds)");
while (nRead)      nRead = SioGetc(SerPort,TICS);
clrscr();
printf("\n\nChecking DCM Receiver transmission.");
printf("\n\nOn front of DCM: Power light (green) ON.");
delay(1500);
printf("\n          Reset/Prgm light (red) OFF.");
printf("\n          TXD (Transmit) RS-232 light (amber) ON.");
delay(1000);
clrscr();
cursor(0,0);
printf("DCM Receiver Ready");
cursor(2,2);
printf("1. Please turn on DCM now then calibrate DCM to 0.");
cursor(2,5);
printf("2. Please check the operation of the footpedal (FTSWT).");
cursor(2,8);
printf("3. Press the Enter key to continue.");
nLookFor = UNFREEZE;
while (nRead != 0xfe)
    {
        nRead = SioGetc(SerPort,TICS);
//      cursor(0,10);
//      printf("10.Get%#4d ",nRead);
//      if (nRead == FREEZE) DoOFFTest();
//      SioError(SioGetc(SerPort,TICS));
        cursor(5,6);
        printf("Footpedal:");
        if (nRead == RIGHT || nRead == LEFT)
            {
                if (nRead == RIGHT)
                    {
                        cursor(25,6);
                        printf("Right");
                    }
                if (nRead == LEFT)
                    {
                        cursor(16,6);
                        printf("Left");
                    }
            }
        else
            {

```

```

    DisplayReadBaud(nRead);
    cursor(16,6);
    clreol();
}
if (kbhit())
{
    nKey=get_char();
    clrscr();
    i=0;
    if (nKey == 'z')
    {
        if (nLookFor == FREEZE)
        {
            nLookFor = UNFREEZE;
            cursor(0,11);
            printf("11.960 ");
            SioError(SioBaud(SerPort,Baud9600));
            nBaud = Baud9600;
            cursor(0,12);
            printf("12.Put ");
            DoONTest();
            click();
        }
        else
        {
            if (nLookFor == UNFREEZE)
            {
                cursor(0,13);
                printf("13.Put%#3d ",nBaud);
                beep();
                DoOFFTest();
                nLookFor = FREEZE;
                cursor(0,14);
                printf("14.115 ");
                SioError(SioBaud(SerPort,Baud115200));
                nBaud = Baud115200;
            }
        }
    }
    while (1)
    {
        nRead = SioGetc(SerPort,TICS);
        cursor(0,15);
        i++;
        printf("15.Read:%#4d For:%#4d P:%#4d K:%#4d i:%#6d",
            nRead, nLookFor, nPut, nKey, i);
    }
}

```

```

        if (i == 1800) break;
        if (nRead == nLookFor) break;
    }
}

if (nKey == 'r')
{
    SioPutc(SerPort,0xbb);
}

if (nKey == 'f')
{
    FreezeTest();
}

if (nKey == 'u')
{
    UnFreezeTest();
}

if (nKey == '9')
{
    cursor(0,12);
    printf("12.960 ");
    SioError(SioBaud(SerPort,Baud9600));
    nBaud = Baud9600;
}

if (nKey == '1')
{
    cursor(0,13);
    printf("13.115 ");
    SioError(SioBaud(SerPort,Baud115200));
    nBaud = Baud115200;
}

if (nKey == 'c')
{
    cursor(0,15);
    printf("15PutON%#4d ON:%x %x %x",
        nRead,SWITCHONCC,SWITCHON55,SWITCHON99);
    DoONTTest();
}

if (nKey == 'v')
{
    cursor(0,16);
    printf("16PutOFF%#4d OFF:%x ",nRead,SWITCHOFFAA);
}

```

```

DoOFFTest();
}

```

```

if (nKey == 'i')
{
    cursor(0,0);
    i=0;
    while (1)
    {
        if (kbhit()) break;
        nRead = SioGetc(SerPort,TICS);
        if (nRead < 6 && nRead)
        {
            printf("%#4d ",nRead);
            i=1;
        }
        else
        {
            if (i && !nRead) printf("\n=%#3d",nRead);
            i = 0;
        }
    }
}

```

```

if (nKey == 'j')
{
    nCount3 = 0;
    nCount4 = 0;
    nCount5 = 0;
    nCount6 = 0;
    nHighCount=0;
    nRead = 0;
    while (1)
    {
        if (kbhit()) break;
        nRead = SioGetc(SerPort,TICS);
        if (nRead == 3) nCount3++;
        if (nRead == 4) nCount4++;
        if (nRead == 5) nCount5++;
        if (nRead == 6) nCount6++;
        nHighCount = nCount3+nCount4+nCount5+nCount6;
        if (nRead > 0) // && nRead < 6)
            printf("\nR=%#3d HC:%#5d 3=%#5d 4=%#5d 5=%#5d

```

```

6=%#5d",

```

```
nRead,nHighCount,nCount3,nCount4,nCount5,nCount6);
```

```
else
```

```
if (!nRead)
```

```
{
```

```
nCount3 = 0;
```

```
nCount4 = 0;
```

```
nCount5 = 0;
```

```
nCount6 = 0;
```

```
nHighCount=0;
```

```
}
```

```
}
```

```
}
```

```
if (nKey == 'k')
```

```
{
```

```
SioPutc(SerPort,0xBB);
```

```
SioDone(SerPort);
```

```
delay(2000);
```

```
SerialSetup();
```

```
}
```

```
if (nKey == 'a')
```

```
{
```

```
nTimes = 0;
```

```
nCounter = 0;
```

```
cNum = 128;
```

```
for (j=0; j<100; j++)
```

```
{
```

```
if (FreezeTest())
```

```
{
```

```
DisplayReadBaud(nRead);
```

```
for (i=0; i<10; i++)
```

```
{
```

```
/*
```

```
cursor(0,0);
```

```
SioError(SioPutc(SerPort,cNum));
```

```
SioError(SioPutc(SerPort,cNum));
```

```
SioError(SioPutc(SerPort,cNum));
```

```
SioError(SioPutc(SerPort,cNum));
```

```
SioError(SioPutc(SerPort,cNum));
```

```
SioError(SioPutc(SerPort,cNum));
```

```
SioError(SioPutc(SerPort,cNum));
```

```
SioError(SioPutc(SerPort,cNum));
```

```
SioError(SioPutc(SerPort,cNum));
```

```
SioError(SioPutc(SerPort,cNum));
```

```
SioError(SioPutc(SerPort,cNum));
```



```

        SioError(SioPutc(SerPort,cNum));
        SioError(SioPutc(SerPort,cNum));*/
        SioPutc(SerPort,cNum);
        SioPutc(SerPort,cNum+1);
        SioPutc(SerPort,cNum+2);
        SioPutc(SerPort,cNum+3);
        SioPutc(SerPort,cNum+4);
        SioPutc(SerPort,cNum+5);
        SioPutc(SerPort,cNum+6);
        SioPutc(SerPort,cNum+7);
        SioPutc(SerPort,cNum+8);
        SioPutc(SerPort,cNum+9);
        SioPutc(SerPort,cNum+10);
        SioPutc(SerPort,cNum+11);
        SioPutc(SerPort,cNum+12);
    }
    clrscr();
    nTimes++;
    cursor(40,9);
    printf("9.A.T: %#7d Baud: %#3d Num: %x",
        nTimes,nBaud,cNum);
    nRead = UnFreezeTest();
    DisplayReadBaud(nRead);
    if (kbhit())
    {
        nKey = get_char();
        if (nKey == '+') cNum++;
        else if (nKey == '-') cNum--;
        else break;
    }
    click();
}

}

if (nKey == UP)
{
    TICS++;
    cursor(40,3);
    printf("TICS: %#4d",TICS);
}
if (nKey == DN)
{
    TICS--;
    cursor(40,3);
    printf("TICS: %#4d",TICS);
}

```

```
    }  
    if (nKey == PGUP)  
    {  
        nDTime++;  
        cursor(55,3);  
        printf("Delay: %#4d", nDTime);  
    }  
    if (nKey == PGDN)  
    {  
        nDTime--;  
        cursor(55,3);  
        printf("Delay: %#4d", nDTime);  
    }  
    if (nKey == ENTER) return;  
    if (nKey == ESC) boogieout();  
}  
}
```

```
/* POINT Module for LISTEN Program */  
/* Copyright 1991-9 by James Hoyt Clark */
```

```
#include <stdio.h>  
#include <stdlib.h>  
#include <io.h>  
#include <fcntl.h>  
#include <sys\types.h>  
#include <sys\stat.h>  
#include <graphics.h>  
#include <time.h>  
#include <dos.h>  
#include <string.h>  
#include <conio.h>  
#include <alloc.h>  
#include <math.h>  
#include "twindow.h"  
#include "keys.h"  
#include "pcxlib.h"
```

```
#define xplace 180  
#define ptlen 69  
#define numpts 54  
#define lcol 210  
#define gcount 25
```

```
extern int reading;  
extern int wavepoint;  
extern int g_mode;  
extern int bkcolor;  
extern int pointcolor;  
extern int postcolor;  
extern int basecolor;  
extern int lowcolor;  
extern int letter2color;  
extern int accentcolor;  
extern long int pointpoint;  
extern char ptname[8];  
extern char ptinfo[53];  
extern char pointflag;  
extern char infoflag;  
extern char tokey;  
extern char fromkey;  
extern char diagram;  
extern char lastanatomy;
```

```
extern char ifgraphic;
extern char iftgraph;
extern char outflag;
extern char outholdflag;
extern int foot; //?????
extern char board;
extern unsigned char voltage;
extern unsigned char max;
extern unsigned char min;
extern unsigned char rise;
extern unsigned char fall;
extern char freqname[7];
extern int CARD;
extern int PORTABLE;
extern int SerPort;
extern int IRDA;
```

```
extern char pnum; /* printer */
```

```
extern char wholename[60];
extern WINDOW *wnd_outhold;
extern char findname[12];
extern char filename[80];
extern void (*helpfunc)();
extern void help();
extern char msg_line[80];
```

```
/*extern struct holdvrecord
```

```
{
    char ptname[8];
    unsigned char voltage;
    char freqname[7];
    int waveshape;
    int y;
    int z;
};
```

```
extern struct holdvrecord holdv[90];*/
```

```
extern struct {
    char last[20];
    char first[20];
    char number[8];
    char nick[20];
    char addr[20];
    char city[20];
```

```

    char state[5];
    char zip[8];
    char ctry[20];
    char phone[11];
    char dt[7];
    char birth[7];
    char doc[9];
    char type[5];
    char sex[2];
    char ethnic[20];
    char amt[6];
    char extra[8];
    char visits;
    char samples;
} client;

WINDOW *wnd_point;
WINDOW *wnd_ptiny;
WINDOW *wnd_viewpt;
WINDOW *wnd_volts;

unsigned char diskbuffer[100];
long int npoint;      /* Number of points */
long int usepoint;    /* Selected point user list */
long int pstart;      /* Point start */
long int pend;        /* Point end */
long int last;
long int savepoint;
unsigned int lastcol,lastrow;
unsigned int col,row;
unsigned int xpos;
unsigned int ypos;
unsigned int lastxpos=10;
unsigned int lastypos=10;
char barmaxflag;
char userptcode;
char listviewflg=0;
char meridianflg=0;
long int usept[1200];
char frompoint[8];

struct pointrecord
{
    char ptname[8]; /* 0-7 */
    unsigned char premax; /* 8 */

```

```

        unsigned char premin;
        unsigned char prerise;
        unsigned char prefall;
        unsigned char postmax;
        unsigned char postmin;
        unsigned char postrise;
        unsigned char postfall; /* 14 */
        char prcode; /* 15 */
        char arrow;
    };
    struct pointrecord point[1200];

/* RESEARCH */

void labelpercent()
{
    char temp[3];
    int j;
    int i=100;
    for (j=0; j<9; j++) {
        itoa(i,temp,10);
        outtextxy(75,j*40+45,temp);
        line(95,j*40+50,100,j*40+50);
        line(95,250,620,250);
        i=i-20;
    }
}

void growth()
{
    int key=0;
    char save_screen[25*80*2];
    int MaxX;
    int MaxColors;
    int xstep,color,top,bottom,spacing,yplace;
    char string[40];
    int i,j;
    int percentchange[gcount];
    int percentplace[gcount];
    int difference[gcount];
    extern void (*helpfunc)();
    extern void help();
    for (i=0; i<gcount; i++) percentplace[i] = 9999;
    j=0;
    for (i=0; i<npoint; i++) {

```

```

        if (point[i].arrow == '<') {
            percentchange[j] = (((point[i].premax + point[i].premin)/2)-50)*2;
            percentplace[j] = i;
            difference[j]=(point[i].premax-50)-(point[i].premin-50);
            j++;
        }
    }

    gettext(1,1,80,25,save_screen);
    setgraphmode(g_mode);
    setbkcolor(EGA_LIGHTGRAY);
    setcolor(DARKGRAY);
    setusercharsize(1,1,1,1);
    settextstyle(SMALL_FONT,HORIZ_DIR,USER_CHAR_SIZE);
    unspace_name();
    outtextxy(5,1,wholename);
    labelpercent();
    settextstyle(SMALL_FONT,VERT_DIR,USER_CHAR_SIZE);
    outtextxy(60,210,"Percent Change");
    rectangle(100,20,620,420);
    xstep=15;
    spacing=5;
    j=110;
    for (i=0; i<gcount; i++) {
        if (percentplace[i] != 9999) {
            settextstyle(SMALL_FONT,HORIZ_DIR,USER_CHAR_SIZE);
            sprintf(string,"%d %d %d",i+1,percentchange[i],difference[i]);
            outtextxy(1,i*10+20,string);
            color = random(MaxColors);
            setfillstyle(i+1,color);
            yplace = 250;
            if (point[percentplace[i]].premax != 0 ) {
                if (percentchange[i] < 0) {
                    top = 250;
                    bottom = 250-(percentchange[i]*2);
                    yplace = bottom;
                }
                else {
                    top = 250-(percentchange[i]*2);
                    bottom = 250;
                }
                bar3d(j,top,j+xstep,bottom,5,1);
                line(j+(xstep/2),top+difference[i],j+(xstep/2),top-difference[i]);
                line(j+(xstep/2-3),top+difference[i],j+(xstep/2+3),top+difference[i]);
                line(j+(xstep/2-3),top-difference[i],j+(xstep/2+3),top-difference[i]);
            }
        }
    }

```

```

        settextstyle(SMALL_FONT, VERT_DIR, USER_CHAR_SIZE);
        sprintf(string, "%s", point[percentplace[i]].ptname);
        outtextxy(j+2, yplace, string);
        j += xstep;
        j += spacing;
    }
}

while (key != ESC) key = get_char();
settext1();
backtext();
puttext(1, 1, 80, 25, save_screen);
}

/* DISK OPS */

/*void saveholdv(int holdpos)
{
    strncpy(holdv[holdpos].ptname, point[usept[pointpoint]].ptname, 8);
    holdv[holdpos].voltage = voltage;
    strcpy(holdv[holdpos].freqname, freqname);
    writeholdvfile();
} */

void selectpoints()
{
    long int i;
    usepoint=0;
    for (i=0; i<npoint; i++) {
        if (point[i].ptcode <= userptcode) {
            usept[usepoint]=i;
            usepoint++;
        }
    }
}

void selectpointsletter(int letter)
{
    long int i;
    usepoint=0;
    for (i=0; i<npoint; i++) {
        if (point[i].ptcode == letter) {
            usept[usepoint]=i;
            usepoint++;
        }
    }
}

```



```
}
```

```
void getpoint(long int position)
```

```
{
```

```
int fhandle,i,j;
```

```
if ((fhandle=open("POINT.DAT",O_RDONLY|O_BINARY)) == -1) {
```

```
    operationinfo("errorop",10,5,iskey);
```

```
    exit(1);
```

```
}
```

```
lseek(fhandle,ptlen*position,SEEK_CUR);
```

```
if ((read(fhandle,diskbuffer,ptlen)) == -1) {
```

```
    operationinfo("errorrd",10,5,iskey);
```

```
    exit(1);
```

```
}
```

```
strncpy(ptname,diskbuffer,8);
```

```
j=0;
```

```
for (i=9; i<62; i++) {
```

```
    ptinfo[j]=diskbuffer[i];
```

```
    j++;
```

```
}
```

```
xpos = (diskbuffer[62] & 0x01)*0x100+diskbuffer[63];
```

```
ypos = diskbuffer[64]*0x100+diskbuffer[65];
```

```
diagram = diskbuffer[66];
```

```
close(fhandle);
```

```
}
```

```
void writeptread(long int position)
```

```
{
```

```
int fhandle;
```

```
long int poslong;
```

```
position=usept[position];
```

```
strncpy(diskbuffer,point[position].ptname,8);
```

```
diskbuffer[8]=point[position].premax;
```

```
diskbuffer[9]=point[position].premin;
```

```
diskbuffer[10]=point[position].prerise;
```

```
diskbuffer[11]=point[position].prefall;
```

```
diskbuffer[12]=point[position].postmax;
```

```
diskbuffer[13]=point[position].postmin;
```

```
diskbuffer[14]=point[position].postrise;
```

```
diskbuffer[15]=point[position].postfall;
```

```
diskbuffer[16]=point[position].ptcode;
```

```
diskbuffer[17]=point[position].arrow;
```

```
if ((fhandle=open("PTREAD.DAT",O_WRONLY|O_BINARY,S_IREAD|S_IWRITE)) == -1) {
```

```
    operationinfo("errorrd",10,5,iskey);
```

```
    exit(1);
```

```

    }
    poslong=18*position;
    lseek(fhandle,poslong,SEEK_CUR);
    if (write(fhandle,diskbuffer,18) == -1) {
        operationinfo("errorwt",10,5,iskey);
        exit(1);
    }
    close(fhandle);
}

```

```

void readpointfile()
{
    FILE *fptr;
    npoint=0;
    if( (fptr=fopen("PTREAD.DAT","rb"))==NULL ) {
        operationinfo("errorfl",10,5,iskey);
        exit(0);
    }
    else {
        while( fread(&point[npoint],sizeof(point[0]),1,fptr)==1 ) npoint++;
        fclose(fptr);
    }
    selectpoints();
}

```

```

void writepointfile()
{
    FILE *fptr;
    if( (fptr=fopen("PTREAD.DAT","wb"))==NULL ) {
        operationinfo("errorfl",10,5,iskey);
        exit(0);
    }
    else {
        fwrite(point,sizeof(point[0]),npoint,fptr);
        fclose(fptr);
    }
}

```

```

void zeropoint(long int j)
{
    point[j].premax=0;
    point[j].premin=0;
    point[j].prerise=0;
    point[j].prefall=0;
    point[j].postmax=0;
}

```

```

point[j].postmin=0;
point[j].postrise=0;
point[j].postfall=0;
}

```

```

void changeptcode()
{
char key;
twobells();
key = get_char();
key = toupper(key);
if (key > '@' && key < '[') {
    point[usept[pointpoint]].ptcode = key;
    writeptread(pointpoint);
}
}

```

```

void nxtpoint()
{
pointpoint++;
if (pointpoint > usepoint-1) pointpoint = 0;
}

```

```

void prevpoint()
{
pointpoint--;
if (pointpoint < 0) pointpoint = usepoint-1;
}

```

```

void showinfopoint()
{
getpoint(usept[pointpoint]);
wcursor(wnd_point,0,20);
if (infoflag) {
    reverse_video(wnd_point);
    wprintf(wnd_point,"%s",ptinfo);
    normal_video(wnd_point);
}
else {
    wprintf(wnd_point,"
");
}
}

```

```

void clearptarrow()
{

```

```

long int i;
for (i=0; i<npoint; i++) point[i].arrow = '';
writepointfile();
}

ptarrowcount()
{
int i,j=0;
for (i=0; i<npoint; i++) if (point[i].arrow == '<') j++;
return j;
}

void onelistpoint(long int j)
{
if (pointpoint == j) {
    showinfopoint();
    lastcol=col;
    lastrow=row;
    reverse_video(wnd_point);
}
wcursor(wnd_point,col,row);
fprintf(wnd_point,"%s%c",point[usept[j]].ptname,point[usept[j]].arrow);
if (pointflag) {
    fprintf(wnd_point,"%#3d %#3d %#3d %#3d",
        point[usept[j]].postmax,point[usept[j]].postmin,
        point[usept[j]].postrise,point[usept[j]].postfall);
}
else fprintf(wnd_point,"%#3d %#3d %#3d %#3d",
    point[usept[j]].premax,point[usept[j]].premin,
    point[usept[j]].prerise,point[usept[j]].prefall);
normal_video(wnd_point);
}

void display_points()
{
long int j;
col=0;
row=1;
display_window(wnd_point);
if (pointpoint > pstart+numpts-1) {
    pstart=pointpoint;
    clear_window(wnd_point);
}
if (pstart+numpts < usepoint) pend = pstart+numpts;
else pend = usepoint;

```

```

for (j=pstart; j<pend; j++) {
    onelistpoint(j);
    col = col+27;
    if (col > 60 ) {
        col=0;
        row++;
    }
}
}

```

```

void showview()
{
    long int j;
    wcursor(wnd_viewpt,0,0);
    for (j=pstart; j<pend; j++) {
        getpoint(usept[j]);
        if (j == pointpoint) reverse_video(wnd_viewpt);
        wprintf(wnd_viewpt,"%s %s %c",ptname,ptinfo,point[usept[j]].ptcode);
        if (pend != j) wprintf(wnd_viewpt,"\n");
        normal_video(wnd_viewpt);
    }
}

```

```

void setpointwindow()
{
    int j;
    showline("msg_PtTitle  ");
    set_title(wnd_point,msg_line);
    set_colors(wnd_point,ALL,pointcolor,WHITE,DIM);
    set_colors(wnd_point,ACCENT,accentcolor,WHITE,BRIGHT);
    display_points();
    if (pointflag) showline("msg_PostMMRF  ");
    else showline("msg_BaseMMRF  ");
    for (j=0; j<3; j++) {
        wcursor(wnd_point,j*27,0);
        wprintf(wnd_point,msg_line);
    }
}

```

```

void displaypoint()
{
    display_window(wnd_ptiny);
    if (pointflag) {
        set_colors(wnd_ptiny, ALL, postcolor, WHITE, DIM);
        wprintf(wnd_ptiny,"\n %s %#3d %#3d %#3d %#3d",

```

```

        point[usept[pointpoint]].ptname,
        point[usept[pointpoint]].postmax,point[usept[pointpoint]].postmin,
        point[usept[pointpoint]].postrise,point[usept[pointpoint]].postfall);
    }
else {
    set_colors(wnd_ptiny, ALL, basecolor, WHITE, DIM);
    wprintf(wnd_ptiny, "\n %s %#3d %#3d %#3d %#3d",
        point[usept[pointpoint]].ptname,
        point[usept[pointpoint]].premax,point[usept[pointpoint]].premin,
        point[usept[pointpoint]].prerise,point[usept[pointpoint]].prefall);
    }
}

```

```

void gosetptiny()
{
    wnd_ptiny = establish_window(42,0,3,27);
    showline("msg_PointMMRF ");
    set_title(wnd_ptiny,msg_line);
    set_colors(wnd_ptiny, ALL, letter2color, WHITE, DIM);
    displaypoint();
}

```

```

void showchange(int aboveORone)
{
    if (aboveORone) selectpointsletter(aboveORone);
    else selectpoints();
    if (listviewflg) showview();
    else {
        clear_window(wnd_point);
        setpointwindow();
    }
    pstart=0;
    pointpoint=0;
}

```

```

void changepoints(void)
{
    int key=0;
    WINDOW *wnd_chanpt;
    setmenuhelp("slpoint",20,14);
    while (key != ESC) {
        wnd_chanpt = establish_window(67,20,5,13);
        showline("msg_Select ");
        set_title(wnd_chanpt,msg_line);
        set_colors(wnd_chanpt,ALL,BLACK,AQUA,DIM);
    }
}

```

```

set_colors(wnd_chanpt,ACCENT,accentcolor,WHITE,BRIGHT);
display_window(wnd_chanpt);
wcursor(wnd_chanpt,1,0);
showline("msg_ListCharactr");
wprintf(wnd_chanpt,msg_line,userptcode);
wcursor(wnd_chanpt,1,1);
showline("msg_UsedPoints ");
wprintf(wnd_chanpt,msg_line,usepoint);
wcursor(wnd_chanpt,0,2);
showline("msg_TotalPoints ");
wprintf(wnd_chanpt,msg_line,npoint);
key = get_char();
key = toupper(key);
delete_window(wnd_chanpt);
if (key > '@' && key < '[') {
    showchange(key);
    userptcode=key;
}
switch(key) {
    case RSWITCH:
    case DN: userptcode++;
        if (userptcode > 'Z') userptcode = 'A';
        showchange(0);
        break;
    case LSWITCH:
    case UP: userptcode--;
        if (userptcode < 'A') userptcode = 'Z';
        showchange(0);
        break;
    default: break;
}
}
}

```

```

void meridian()
{
    char temp[3];
    long int i;
    temp[2]=0;
    savepoint=pointpoint;
    strncpy(temp,point[userpt[pointpoint]].ptname,2);
    usepoint=0;
    for (i=0; i<npoint; i++)
        if (strcmp(temp,point[i].ptname,2) == 0) {
            usept[usepoint]=i;

```

```

        usepoint++;
    }

    pstart=0;
    pointpoint=0;
}

void pointbarheading(void)
{
    setviewport(0,10,178,18,1);
    clearviewport();
    setcolor(pointcolor);
    if (barmaxflag) showline("msg_PointMMMM ");
    else showline("msg_PointRFRF ");
    outtextxy(3,0,msg_line);
}

void display_pointbars(void)
{
    long int j,k;
    long int linenum;
    int maxlen,minlen;
    char string[25];
    char string1[24];

    if (pstart+40 < usepoint) pend = pstart+40;
    else pend = usepoint;
    linenum = 0;
    for (j=pstart; j<pend; j++) {
        k=(linenum+2)*10;
        if (j == pointpoint) {
            setcolor(bkcolor);
            rectangle(1,last,xplace-1,last+10);
            setviewport(50,k,xplace-1,k+10,1);
            clearviewport();
            setviewport(0,0,639,479,1);
            setcolor(lowcolor);
            rectangle(1,k,xplace-1,k+10);
            last=k;
        }
        setcolor(pointcolor);
        setttextjustify(LEFT_TEXT,TOP_TEXT);
        setviewport(0,0,639,479,1);
        if (barmaxflag) {
            maxlen = point[usept[j]].premax*4+xplace;
            minlen = point[usept[j]].premin*4+xplace;

```



```

        setfillstyle(SOLID_FILL,RED);
        bar(xplace,k,minlen,k+3);
        setfillstyle(SLASH_FILL,BLUE);
        bar(minlen,k,maxlen,k+3);
        setfillstyle(SOLID_FILL,bkcolor);
        bar(maxlen,k,xplace+400,k+3);
        k=k+4;
        maxlen = point[usept[j]].postmax*4+xplace;
        minlen = point[usept[j]].postmin*4+xplace;
        setfillstyle(SOLID_FILL,BLUE);
        bar(xplace,k,minlen,k+3);
        setfillstyle(SLASH_FILL,RED);
        bar(minlen,k,maxlen,k+3);
        setfillstyle(SOLID_FILL,bkcolor);
        bar(maxlen,k,xplace+400,k+3);
        sprintf(string, "%s%#3d %>#3d %>#3d %>#3d",
        point[usept[j]].ptname,
        point[usept[j]].premax,point[usept[j]].premin,
        point[usept[j]].postmax,point[usept[j]].postmin);
    }
else {
        maxlen = point[usept[j]].prerise*4+xplace;
        minlen = point[usept[j]].prefall*4+xplace;
        setfillstyle(SOLID_FILL,YELLOW);
        bar(xplace,k,minlen,k+3);
        setfillstyle(SLASH_FILL,GREEN);
        bar(minlen,k,maxlen,k+3);
        setfillstyle(SOLID_FILL,bkcolor);
        bar(maxlen,k,xplace+400,k+3);
        k=k+4;
        maxlen = point[usept[j]].postrise*4+xplace;
        minlen = point[usept[j]].postfall*4+xplace;
        setfillstyle(SOLID_FILL,GREEN);
        bar(xplace,k,minlen,k+3);
        setfillstyle(SLASH_FILL,YELLOW);
        bar(minlen,k,maxlen,k+3);
        setfillstyle(SOLID_FILL,bkcolor);
        bar(maxlen,k,xplace+400,k+3);
        sprintf(string, "%s%#3d %>#3d %>#3d %>#3d",
        point[usept[j]].ptname,
        point[usept[j]].prerise,point[usept[j]].prefall,
        point[usept[j]].postrise,point[usept[j]].postfall);
    }
    outtextxy(4,(linenum+2)*10+2,string);
    linenum++;

```

```

    }
    line(380,20,380,420);
}

void display1pointbar()
{
    long int k,linenum;
    int maxlen,minlen;
    char string[25];

    linenum = pointpoint-pstart;
    k=(linenum+2)*10;
    setcolor(bkcolor);
    setviewport(50,k,xplace-1,k+10,1);
    clearviewport();
    setviewport(0,0,639,479,1);
    setcolor(lowcolor);
    rectangle(1,k,xplace-1,k+10);
    last=k;
    if (barmaxflag) {
        maxlen = point[usept[pointpoint]].premax*4+xplace;
        minlen = point[usept[pointpoint]].premin*4+xplace;
        setfillstyle(SOLID_FILL,RED);
        bar(xplace,k,minlen,k+3);
        setfillstyle(SLASH_FILL,BLUE);
        bar(minlen,k,maxlen,k+3);
        setfillstyle(SOLID_FILL,bkcolor);
        bar(maxlen,k,xplace+400,k+3);
        k=k+4;
        maxlen = point[usept[pointpoint]].postmax*4+xplace;
        minlen = point[usept[pointpoint]].postmin*4+xplace;
        setfillstyle(SOLID_FILL,BLUE);
        bar(xplace,k,minlen,k+3);
        setfillstyle(SLASH_FILL,RED);
        bar(minlen,k,maxlen,k+3);
        setfillstyle(SOLID_FILL,bkcolor);
        bar(maxlen,k,xplace+400,k+3);
        sprintf(string,"%s%#3d %#3d %#3d %#3d",
            point[usept[pointpoint]].ptname,point[usept[pointpoint]].premax,
            point[usept[pointpoint]].premin,point[usept[pointpoint]].postmax,
            point[usept[pointpoint]].postmin);
    }
    else {
        maxlen = point[usept[pointpoint]].prerise*4+xplace;
        minlen = point[usept[pointpoint]].prefall*4+xplace;
    }
}

```

```

        setfillstyle(SOLID_FILL,YELLOW);
        bar(xplace,k,minlen,k+3);
        setfillstyle(SLASH_FILL,GREEN);
        bar(minlen,k,maxlen,k+3);
        setfillstyle(SOLID_FILL,bkcolor);
        bar(maxlen,k,xplace+400,k+3);
        k=k+4;
        maxlen = point[usept[pointpoint]].postrise*4+xplace;
        minlen = point[usept[pointpoint]].postfall*4+xplace;
        setfillstyle(SOLID_FILL,GREEN);
        bar(xplace,k,minlen,k+3);
        setfillstyle(SLASH_FILL,YELLOW);
        bar(minlen,k,maxlen,k+3);
        setfillstyle(SOLID_FILL,bkcolor);
        bar(maxlen,k,xplace+400,k+3);
        sprintf(string, "%s%#3d %#3d %#3d %#3d",
        point[usept[pointpoint]].ptname,point[usept[pointpoint]].prerise,
        point[usept[pointpoint]].prefall,point[usept[pointpoint]].postrise,
        point[usept[pointpoint]].postfall);
    }
    setcolor(pointcolor);
    outtextxy(4,(linenum+2)*10+2,string);
    line(380,20,380,420);
}

void showinfograph()
{
    char string[53];
    if (infoflag) {
        getpoint(usept[pointpoint]);
        setviewport(0,0,639,479,1);
        setcolor(pointcolor);
        rectangle(1,442,420,456);
        setviewport(2,443,419,455,1);
        clearviewport();
        sprintf(string,"%s",ptinfo);
        outtextxy(4,2,string);
        setviewport(0,0,639,479,1);
    }
    else {
        setviewport(1,442,420,456,1);
        clearviewport();
    }
}

```

```

void displaypointgraph()
{
    char string[35];
    int fillcolor;
    getpoint(usept[pointpoint]);
    setcolor(pointcolor);
    setviewport(0,0,639,479,1);
    rectangle(40,426,231,437);
    setviewport(0,4,290,421,1);
    setfillstyle(SOLID_FILL,BLACK);
    settextrjustfy(LEFT_TEXT,TOP_TEXT);
    if (diagram != lastanatomy) dopicture(diagram);
    else fillellipse(lastxpos,lastypos,3,3);
    if (diskbuffer[62] > 1) fillcolor=EGA_RED;
    else fillcolor=EGA_WHITE;
    setfillstyle(SOLID_FILL,fillcolor);
    fillellipse(xpos,ypos,3,3);
    setcolor(pointcolor);
    setviewport(41,427,230,436,1);
    clearviewport();
    sprintf(string,"%s %#3d %#3d %#3d %#3d",
    point[usept[pointpoint]].ptname,
    point[usept[pointpoint]].premax,point[usept[pointpoint]].premin,
    point[usept[pointpoint]].postmax,point[usept[pointpoint]].postmin);
    outtextxy(4,2,string);
    setviewport(0,0,639,479,1);
    lastanatomy=diagram;
    lastxpos=xpos;
    lastypos=ypos;
    showinfofgraph();
}

```

```

void pointhome()
{
    long int i;
    for (i=pointpoint; i>0; i--) {
        getpoint(usept[i]);
        if (diagram != lastanatomy) break;
        pointpoint=i;
    }
    displaypointgraph();
}

```

```

void pointend()
{

```

```

long int i;
for (i=pointpoint; i<usepoint; i++) {
    getpoint(usept[i]);
    if (diagram != lastanatomy) break;
    pointpoint=i;
}
displaypointgraph();
}

```

```

void storepre()
{
point[usept[pointpoint]].premax = max;
point[usept[pointpoint]].premin = min;
point[usept[pointpoint]].prerise = rise;
point[usept[pointpoint]].prefall = fall;
}

```

```

void storepost()
{
point[usept[pointpoint]].postmax = max;
point[usept[pointpoint]].postmin = min;
point[usept[pointpoint]].postrise = rise;
point[usept[pointpoint]].postfall = fall;
}

```

```

void clearbars()
{
setviewport(1,20,580,420,1);
clearviewport();
setviewport(0,0,639,479,1);
}

```

```

void labelbarbox()
{
int j;
char string[4];
setviewport(0,0,639,479,1);
rectangle(0,19,581,421);
for (j=0; j<5; j++) {
    itoa(j*25,string,10);
    outtextxy(j*97+xplace,11,string);
    outtextxy(j*97+xplace,423,string);
}
}

```

```

void switchpoint()
{
long int j;
for (j=0; j<usepoint; j++) {
    if (strcmp(frompoint,point[usept[j]].ptname) == 0) {
        pointpoint=j;
        beep();
        return;
    }
}

```

```

makesound(50,3);
}

```

```

void switchtooth(char direction)

```

```

{
static int updown[32] = {32,31,30,29,28,27,26,25,24,23,22,21,20,19,18,17,
                        16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1};
static int lefrit[32] = {16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,
                        32,31,30,29,28,27,26,25,24,23,22,21,20,19,18,17};

char ascii[3];
int number;
strcpy(frompoint,point[usept[pointpoint]].ptname);
ascii[0]=point[usept[pointpoint]].ptname[1];
ascii[1]=point[usept[pointpoint]].ptname[2];
ascii[2]=0;
number=atoi(ascii);
if (direction == 'U') number = updown[number-1];
else number = lefrit[number-1];
itoa(number,ascii,10);
frompoint[1]=ascii[0];
if (number < 10) frompoint[2] = ' ';
else frompoint[2] = ascii[1];
switchpoint();
}

```

```

void otherpoint()

```

```

{
char len;
strcpy(frompoint,point[usept[pointpoint]].ptname);
if (diagram == 'M') {
    switchtooth('O');
    return;
}
if (ptname[0] == 'P' && ptname[1] == 'A') {
    frompoint[0] = 'S';
}

```

```

        frompoint[1] = 'P';
    }
    else if (ptname[1] == 'P') {
        frompoint[0] = 'P';
        frompoint[1] = 'A';
    }

    len=6;
    if (point[usept[pointpoint]].ptname[6] == ' ') len=5;
    if (point[usept[pointpoint]].ptname[5] == ' ') len=4;
    if (point[usept[pointpoint]].ptname[4] == ' ') len=3;
    if (point[usept[pointpoint]].ptname[len] == 'R') frompoint[len]='L';
    else frompoint[len]='R';
    switchpoint();
}

```

```

void foothand()
{
    char j;
    getpoint(usept[pointpoint]);
    j = diagram;
    j=toupper(j);
    if (j == 'H') strcpy(frompoint, "FICR ");
    if (j == 'F') strcpy(frompoint, "PCCR ");
    if (j == 'P') strcpy(frompoint, "FIC_R ");
    if (j == 'Q') strcpy(frompoint, "PCC_R ");
    if (j == 'M') {
        switchtooth('U');
        return;
    }
    switchpoint();
}

```

```

void vbars()
{
    char save_screen[25*80*2];
    int MaxX;
    int MaxColors;
    long int extpoint[10];
    char extvalue[10];
    char str[40];
    unsigned char prtstatus;
    struct viewporttype vp;
    int i,j;
    int xstep, ystep,color,bheight,hdiff;
    char buffer[4];
}

```

```

char h,maxs[2000];
int key=0;
j=0;
for (i=0; i<npoint; i++) {
    h=point[i].premax;
    if (h==100) h=1;
    if (h > 50) h=100-h;
    maxs[j]=h;
    j++;
}
for (i=0; i<10; i++) {
    extpoint[i]=3999;
    extvalue[i]=99;
}
for (j=0; j<10; j++) {
    for (i=6; i<npoint; i++) { /*ignore first 6 */
        if (maxs[i] != 0) {
            if (maxs[i] < extvalue[j]) {
                extpoint[j]=i;
                extvalue[j]=maxs[i];
            }
        }
    }
    maxs[extpoint[j]]=0;
}
gettext(1,1,80,25,save_screen);
setgraphmode(g_mode);
Bibox(1,0,0);
setbkcolor(bkcolor);
MaxX=getmaxx();
MaxColors=getmaxcolor()+1;
setcolor(MaxColors-1);
getviewsettings(&vp);
settextjustify(CENTER_TEXT,TOP_TEXT);
settextstyle(TRIPLEX_FONT,HORIZ_DIR,3);
showline("msg_Extremes  ");
outtextxy(MaxX/2,16,msg_line);
settext1();
h = 3 * textheight("H");
setviewport(vp.left+50, vp.top+40, vp.right-50, vp.bottom-10, 1);
getviewsettings( &vp );
line(h,h,h,vp.bottom-vp.top-h);
line(h,(vp.bottom-vp.top)-h, (vp.right-vp.left)-h, (vp.bottom-vp.top)-h);
xstep = ((vp.right-vp.left) - (2*h)) / 10;
ystep = ((vp.bottom-vp.top) - (2*h)) / 5;

```



```

j = h;
randomize();
for(i=0; i<10; ++i) {
    color = random(MaxColors);
    setfillstyle(i+1,color);
    if (extpoint[i] != 3999) {
        hdiff=2;
        if (fmod(i,2) == 0) hdiff=-8;
        outtextxy(j,(vp.bottom-vp.top)+hdiff-(h/2),
            point[extpoint[i]].ptname);
        if( i != 10 ){
            bheight = (vp.bottom-vp.top) - h - 1;
            bar3d(j,(vp.bottom-vp.top-h)-(point[extpoint[i]].premax*3.8),
                j+xstep,bheight,15,1);
        }
        j += xstep;
    }
}
j = (vp.bottom-vp.top) - h;
for (i=0; i<11; ++i) {
    line(20,j,513,j);
    itoa(i*10,buffer,10);
    outtextxy(0,j,buffer);
    j -= ystep/2;
}
j=0;
key = get_char();
key=toupper(key);
if (key == 'P') {
    prtstatus=biosprint(2,0,pnum);
    if (prtstatus==0x90) {
        for (j=0; j<10; j++) {
            printf(str,"%s %#3d",
                point[extpoint[j]].ptname,point[extpoint[j]].premax);
            toprinter(str);
        }
        formfeed();
    }
}
ifgraphic=0;
backtext();
puttext(1,1,80,25,save_screen);
}

```

```

void statisticheading(int col,int row)
{
    setcolor(BLUE);
    showline("msg_StatHeading ");
    outtextxy(col*200+52,row*10+38,msg_line);
}

void statistics(void)
{
    char save_screen[25*80*2];
    char str[50];
    unsigned char prtstatus;
    struct viewporttype vp;
    int i,j;
    int row=0;
    int col=0;
    double mean=0;
    long int meanspoint[100];
    long int tempmeans;
    long int tempmeanspoint;
    double means[100];
    double maxmean=0;
    double minmean=0;
    double rismean=0;
    double falmean=0;
    double meandeviation=0;
    double variance=0;
    double stdvariation;
    double diff;
    int key;
    int counter=-1;
    int totalcounter=-1;
    int linecount=1;
    char meanyet=1;
    char lowyet=1;
    gettext(1,1,80,25,save_screen);
    setgraphmode(g_mode);
    setbkcolor(bkcolor);
    settextjustify(LEFT_TEXT,TOP_TEXT);
    settextstyle(TRIPLEX_FONT,HORIZ_DIR,2);
    showline("msg_Statistics ");
    outtextxy(10,5,msg_line);
    settextl();
    statisticheading(col,row);
    for (i=6; i<npoint; i++) {

```

```

        if (point[i].premax > 0) {
            counter++;
            means[counter] = point[i].premax;
            meanspoint[counter]=i;
            mean = mean+point[i].premax;
        }
        if (counter == 99) break;
    }
    counter++;
    if (counter != 0) maxmean=mean/counter;
    for (i=0; i<counter; i++) {
        for (j=0; j<counter; j++) {
            if (means[i] > means[j]) {
                tempmeans=means[i];
                tempmeanspoint=meanspoint[i];
                means[i]=means[j];
                meanspoint[i] = meanspoint[j];
                means[j]=tempmeans;
                meanspoint[j]=tempmeanspoint;
            }
        }
    }
    for (i=0; i<counter; i++) {
        if ((diff=means[i]-maxmean) > 0) setcolor(RED);
        if (diff < 0) {
            setcolor(YELLOW);
            if (lowyet) {
                if (row != 0) row++;
                if (row == 40) {
                    col++;
                    row=0;
                    statisticheading(col,row);
                }
                linecount=1;
            }
            lowyet=0;
        }
        if (diff == 0) {
            setcolor(BLUE);
            if (meanyet) {
                if (row != 0) row++;
                if (row == 40) {
                    col++;
                    row=0;
                    statisticheading(col,row);
                }
            }
        }
    }
}

```

```

        }
        linecount=1;
    }
    meanyet=0;
}
meandeviation=meandeviation+abs(diff);
variance=variance+(diff*diff);
sprintf(str,"%#3d %s %#3d %+.2f",
        linecount,point[meanspoint[i]].ptname,
        point[meanspoint[i]].premax,diff);
outtextxy(col*200+20,row*10+50,str);
row++;
if (row == 40) {
    col++;
    row=0;
    statisticheading(col,row);
}
linecount++;
}
mean=0;
for (i=6; i<npoint; i++) {
    if (point[i].premax > 0) {
        totalcounter++;
        mean = mean+point[i].premin;
    }
}
if (totalcounter != -1) minmean=mean/(totalcounter+1);
totalcounter=-1;
mean=0;
for (i=6; i<npoint; i++) {
    if (point[i].premax > 0) {
        totalcounter++;
        mean = mean+point[i].prerise;
    }
}
if (totalcounter != -1) risemean=mean/(totalcounter+1);
totalcounter=-1;
mean=0;
for (i=6; i<npoint; i++) {
    if (point[i].premax > 0) {
        totalcounter++;
        mean = mean+point[i].prefall;
    }
}
if (totalcounter != -1) falmean=mean/(totalcounter+1);

```

```

meandeviation=meandeviation/(counter-1);
variance=variance/(counter-1);
setviewport(400,268,639,479,1);
setcolor(BLUE);
rectangle(0,0,237,209);
setfillstyle(SOLID_FILL,RED);
bar(3,3,234,206);
showlinenum("msg_StatSummary ",100,12,10);
setcolor(WHITE);
showlinenum("msg_StatPoints ",counter,30,32);
showlinenum("msg_TestPoints ",totalcounter+1,54,44);

showlinereal("msg_MAXMean ",maxmean,94,72);
showlinereal("msg_MeanDeviation",meandeviation,46,84);
stdvariation=sqrt(variance);
showlinereal("msg_StdDeviation",stdvariation,14,96);
showlinereal("msg_MAXVariance ",variance,94,108);

showlinereal("msg_MINMean ",minmean,94,160);
showlinereal("msg_RISMean ",rismean,94,175);
showlinereal("msg_FALMean ",falmean,94,190);

key = get_char();
key=toupper(key);
if (key == 'P') {
    prtstatus=biosprint(2,0,pnum);
    if (prtstatus==0x90) {
        printf("\n");
        sprintf(str,"%#3d %#3d %#3d %#3d",maxmean,minmean,rismean,falmean);
        toprinter(str);
        formfeed();
    }
}
ifgraphic=0;
backtext();
puttext(1,1,80,25,save_screen);
}

void pointbars()
{
int key=0;
char save_screen[25*80*2];
extern void (*helpfunc)();
extern void help();
void barsmenu();

```

```

barmaxflag=1;
helpfunc=barsmenu;
gettext(1,1,80,25,save_screen);
setgraphmode(g_mode);
setbkcolor(bkcolor);
labelbarbox();
labelbar();
showline("msg_BasePost  ");
outtextxy(72,0,msg_line);
last = 40;
pstart=pointpoint;
clearbars();
display_pointbars();
pointbarheading();
iftgraph = 0;
ifgraphic = 1;
while (key != ESC) {
    if (PORTABLE == KEYPORT) outputholdPort();
    if (CARD == KEYCARD) outputhold();
    INPUT();
    foot=footpedal();
    if(kbhit() || foot > 0) {
        if (foot>0) {
            key=foot;
        }
        else {
            key = get_char();
            key = toupper(key);
        }
        switch(key) {
            case LSWITCH:
            case UP: prevpoint();
                if (pointpoint < pstart+1 || pointpoint > pend) {
                    pstart=pointpoint-39;
                    clearbars();
                    if (pstart < 0) {
                        pstart=0;
                        pointpoint=0;
                    }
                }
                display_pointbars();
                break;
            case RSWITCH:
            case DN: nxtpoint();
                if (pointpoint > pstart+39 || pointpoint == 0) {

```

```

        pstart = pointpoint;
        clearbars();
    }
    display_pointbars();
    break;
case PGUP: pstart=pstart-40;
    if (pstart < 0) pstart=0;
    clearbars();
    pointpoint=pstart+39;
    display_pointbars();
    break;
case PGDN: pstart = pstart+40;
    if (pstart > usepoint+1) pstart=0;
    clearbars();
    pointpoint=pstart;
    display_pointbars();
    break;
case 'H': if (!pointflag) break;
    toggle_outhold();
    outholdgraph();
    break;
case HOME: pointpoint=pstart;
    clearbars();
    display_pointbars();
    break;
case END: pointpoint=pend-1;
    clearbars();
    display_pointbars();
    break;
case CTRL_HOME: pstart=0;
    pointpoint=0;
    clearbars();
    display_pointbars();
    break;
case CTRL_END: pstart=usepoint-40;
    if (pstart < 0) pstart=0;
    pointpoint=usepoint-1;
    clearbars();
    display_pointbars();
    break;
case 'O': otherpoint();
    clearbars();
    pstart=pointpoint;
    display_pointbars();
    break;

```

```

        case 'X': if (barmaxflag) barmaxflag=0;
                    else barmaxflag=1;
                    clearbars();
                    display_pointbars();
                    pointbarheading();
                    break;
        case '*': if (pointflag) {
                            setbase();
                            outholdgraph();
                        }
                    else setpost();
                    display_pointbars();
                    pointbarheading();
                    break;
        case ESC: break;
        default: break;
    }
}

backtext();
puttext(1,1,80,25,save_screen);
display_points();
iftgraph = 1;
ifgraphic = 0;
}

void jumptopoint()
{
    long int i,j;
    int len;
    jumpname();
    len=strlen(findname);
    if (len==0) return;
    for (i=0; i<npoint; i++) {
        if (strcmp(findname,point[i].ptname,len) == 0) break;
    }
    if (i < npoint) {
        if (point[i].ptcode != userptcode) {
            userptcode = point[i].ptcode;
            selectpoints();
        }
        for (j=0; j<usepoint; j++) if (usept[j] == i) break;
        pointpoint=j;
        pstart=j;
        if (fromkey == 'P') {

```



```

        clear_window(wnd_point);
        setpointwindow();
    }
    else displaypoint();
}
else makesound(20,10);
}

void setforpoint()
{
    fromkey='P';
    tokey='P';
    iftgraph=1;
}

void pointlist()
{
    setforpoint();
    setbase();
    setmenuhelp("point ",3,1);
    wnd_point = establish_window(0,0,23,79);
    showline("msg_PtTitle ");
    set_title(wnd_point,msg_line);
    setpointwindow();
    setreadwindow();
    gosetouthold();
    ifgraphic=0;
}

void viewpoint()
{
    int key=0;
    long int j;

    listviewflg=1;
    setmenuhelp("morept ",4,2);
    wnd_viewpt = establish_window(0,0,25,67);
    showline("msg_PointInforma");
    set_title(wnd_viewpt,msg_line);
    set_colors(wnd_viewpt,ALL,BLACK,AQUA,DIM);
    set_colors(wnd_viewpt,ACCENT,accentcolor,WHITE,BRIGHT);
    display_window(wnd_viewpt);
    pstart=pointpoint;
    while (key != ESC) {
        if (pstart+22 < usepoint) pend = pstart+22;
    }
}

```

```

else pend = usepoint;
showview();
key = get_char();
key = toupper(key);
switch(key) {
    case RSWITCH:
    case DN: pointpoint++;
        if (pointpoint > usepoint-1) {
            pointpoint=0;
            pstart=0;
            clear_window(wnd_viewpt);
        }
        if (pointpoint > pstart+21) {
            pstart = pointpoint;
            clear_window(wnd_viewpt);
        }
        break;
    case LSWITCH:
    case UP: pointpoint--;
        if (pointpoint < 0) {
            pointpoint=usepoint-1;
            pstart=usepoint-22;
            if (pstart < 0) pstart=0;
        }
        if (pointpoint < pstart) {
            pstart=pointpoint-21;
            clear_window(wnd_viewpt);
            if (pstart < 0) {
                pstart=0;
                pointpoint=0;
            }
        }
        break;
    case PGUP: pstart=pstart-22;
        if (pstart < 0) pstart=0;
        clear_window(wnd_viewpt);
        pointpoint=pstart+21;
        break;
    case PGDN: pstart = pstart+22;
        if (pstart > usepoint) pstart=0;
        clear_window(wnd_viewpt);
        pointpoint=pstart;
        break;
    case 'O': otherpoint();
        pstart=pointpoint;

```

```

        break;
    case 'U': foothand();
        pstart=pointpoint;
        break;
    case HOME: pointpoint=pstart;
        break;
    case END: pointpoint=pend-1;
        break;
    case CTRL_HOME: pstart=0;
        pointpoint=0;
        break;
    case CTRL_END: pstart=usepoint-22;
        if (pstart < 0) pstart=0;
        pointpoint=usepoint-1;
        break;
    case 'W': display_time(); break;
    case INS: changeptcode();
        break;
    case 'S': changepoints();
        setmenuhelp("morept ",4,2);
        break;
    default: break;
}
}
delete_window(wnd_viewpt);
}

```

```

void reviewgraph()
{
    struct stat info;
    char save_screen[25*80*2];
    char string[40];
    int key=0;
    int i,j,start,expand,oldexpand,spacing;
    extern void (*helpfunc)();
    extern void help();
    int savpremax;
    int savpremin;
    int savprerise;
    int savprefall;
    gettext(1,1,80,25,save_screen);
    setgraphmode(g_mode);
    setbkcolor(EGA_LIGHTGRAY);
    setcolor(DARKGRAY);
    setusercharsize(1,1,1,1);
}

```

```

settextstyle(SMALL_FONT,HORIZ_DIR,USER_CHAR_SIZE);
showline("msg_Visits  ");
outtextxy(lcol+160,445,msg_line);
outtextxy(5,1,wholename);
labelbar();
sprintf(string,"%s",point[usept[pointpoint]].ptname);
outtextxy(5,11,string);
rectangle(lcol-10,14,608,430);
line(lcol-10,220,608,220);
showline("msg_MMRFDate  ");
sprintf(string,msg_line);
outtextxy(1,30,string);
if (client.visits > 20) start=client.visits-20;
else start=0;
j=0;
spacing=(400/(client.visits-start));
oldexpand=spacing*j+lcol;
readpoint(start);
savpremax=415-(point[usept[pointpoint]].premax*4.01);
savpremin=415-(point[usept[pointpoint]].premin*4.01);
savprerise=415-(point[usept[pointpoint]].prerise*4.01);
savprefall=415-(point[usept[pointpoint]].prefall*4.01);
for (i=start; i<client.visits; i++) {
    readpoint(i);
    if(stat(filename,&info) != 0) {
        operationinfo("Errorst",10,5,iskey);
        break;
    }
    expand=spacing*j+lcol;
    setcolor(DARKGRAY);
    sprintf(string,"%#2d.",i+1);
    outtextxy(1,j*20+40,string);
    setcolor(RED);
    sprintf(string,"%#3d",point[usept[pointpoint]].premax);
    outtextxy(20,j*20+40,string);
    outtextxy(expand,415-(point[usept[pointpoint]].premax*4.01),"*");
    line(oldexpand,savpremax+7,expand,422-(point[usept[pointpoint]].premax*4.01));
    setcolor(BLUE);
    sprintf(string,"%#3d",point[usept[pointpoint]].premin);
    outtextxy(47,j*20+40,string);
    outtextxy(expand,415-(point[usept[pointpoint]].premin*4.01),"#");
    line(oldexpand,savpremin+7,expand,422-(point[usept[pointpoint]].premin*4.01));
    setcolor(MAGENTA);
    sprintf(string,"%#3d",point[usept[pointpoint]].prerise);
    outtextxy(77,j*20+40,string);
}

```

```

        outtextxy(expand,415-(point[usept[pointpoint]].prerise*4.01),"r");
        line(oldexpand,savprerise+7,expand,422-(point[usept[pointpoint]].prerise*4.01));
        setcolor(GREEN);
        sprintf(string,"%#3d",point[usept[pointpoint]].prefall);
        outtextxy(104,j*20+40,string);
        outtextxy(expand,415-(point[usept[pointpoint]].prefall*4.01),"f");
        line(oldexpand,savprefall+7,expand,422-(point[usept[pointpoint]].prefall*4.01));
        setcolor(DARKGRAY);
        strcpy(string,ctime(&info.st_atime));
        outtextxy(55,j*20+50,string);
        itoa(j+1,string,10);
        outtextxy(expand,435,string);
        j++;
        savpremax=415-(point[usept[pointpoint]].premax*4.01);
        savpremin=415-(point[usept[pointpoint]].premin*4.01);
        savprerise=415-(point[usept[pointpoint]].prerise*4.01);
        savprefall=415-(point[usept[pointpoint]].prefall*4.01);
        oldexpand=expand;
    }
while (key != ESC) key = get_char();
settext1();
backtext();
puttext(1,1,80,25,save_screen);
}

void reviewvisits()
{
    struct stat info;
    WINDOW *wnd_visit;
    int i,j,start,key;
    char date[30];
    struct savepoint
    {
        char premax;
        char premin;
        char prerise;
        char prefall;
        char postmax;
        char postmin;
        char postrise;
        char postfall;
    };
    struct savepoint savept[1200];

    setmenuhelp("graphvi",4,2);

```

```

wnd_visit = establish_window(0,0,25,67);
showline("msg_PointVReview");
set_title(wnd_visit,msg_line);
set_colors(wnd_visit,ALL,BLACK,AQUA,DIM);
set_colors(wnd_visit,ACCENT,accentcolor,WHITE,BRIGHT);
display_window(wnd_visit);
wcursor(wnd_visit,1,0);
wprintf(wnd_visit," %s",point[usept[pointpoint]].ptname);
unspace_name();
wprintf(wnd_visit," %s",wholename);
wcursor(wnd_visit,7,2);
showline("msg_DateMMIDRF ");
wprintf(wnd_visit,msg_line);
if (client.visits > 20) start=client.visits-20;
else start=0;
for (i=0; i<npoint; i++) {
    savept[i].premax=point[i].premax;
    savept[i].premin=point[i].premin;
    savept[i].prerise=point[i].prerise;
    savept[i].prefall=point[i].prefall;
    savept[i].postmax=point[i].premax;
    savept[i].postmin=point[i].premin;
    savept[i].postrise=point[i].prerise;
    savept[i].postfall=point[i].prefall;
}
j=0;
for (i=start; i<client.visits; i++) {
    readpoint(i);
    if(stat(filename,&info) != 0) {
        operationinfo("errorst",10,5,yeskey);
        break;
    }
    wcursor(wnd_visit,4,j+3);
    wprintf(wnd_visit,"%#2d. ",i+1);
    strcpy(date,ctime(&info.st_atime));
    date[24] = 0;
    date[25] = 0;
    date[26] = 0;
    wprintf(wnd_visit,"%s ",date);
    wprintf(wnd_visit,"%#3d %#3d %#3d %#3d %#3d",
    point[usept[pointpoint]].premax,point[usept[pointpoint]].premin,
    point[usept[pointpoint]].premax-point[usept[pointpoint]].premin,
    point[usept[pointpoint]].prerise,point[usept[pointpoint]].prefall);
    j++;
}

```

```

while (key != ESC) {
    key = get_char();
    key = toupper(key);
    switch(key) {
        case 'G': if (client.visits > 0) reviewgraph(); break;
        default: break;
    }
}
for (i=0; i<npoint; i++) {
    point[i].premax=savept[i].premax;
    point[i].premin=savept[i].premin;
    point[i].prerise=savept[i].prerise;
    point[i].prefall=savept[i].prefall;
    point[i].premax=savept[i].postmax;
    point[i].premin=savept[i].postmin;
    point[i].prerise=savept[i].postrise;
    point[i].prefall=savept[i].postfall;
}
delete_window(wnd_visit);
}

```

```

void showvolts(void)
{
    float number;
    outportb(SETVOLT,voltage);
    number=voltage;
    wprintf(wnd_volts,"\n %#1.2f",number*.019608);
}

```

```

void tellvolts(void)
{
    wnd_volts = establish_window(32,24,3,9);
    showline("msg_Voltage ");
    set_title(wnd_volts,msg_line);
    set_colors(wnd_volts,ALL,WHITE,BLACK,DIM);
    display_window(wnd_volts);
    showvolts();
}

```

```

void points()
{
    int key=0;
    char washold;
    pstart=pointpoint;
    pointlist();
}

```

```

if (!board) tellvolts();
while (key != ESC) {
    cursoroff();
    if (PORTABLE == KEYPORT) outputholdPort();
    if (CARD == KEYCARD) outputhold();
    INPUT();
    foot = footpedal();
    if (kbhit() || foot>0) {
        if (foot>0) key=foot;
        else {
            key = get_char();
            key = toupper(key);
        }
        switch(key) {
            case UP: if (pointpoint < pstart+3)
                pointpoint = pointpoint+pend-pstart-3;
                else pointpoint = pointpoint-3;
                display_points();
                break;
            case DN: if (pointpoint > pend-4)
                pointpoint = pointpoint+pstart-pend+3;
                else pointpoint = pointpoint+3;
                display_points();
                break;
            case LSWITCH:
            case BS: prevpoint();
                if (pointpoint < pstart) {
                    pstart=pointpoint-53;
                    if (pstart < 0) {
                        pstart=0;
                        pointpoint=0;
                        clear_window(wnd_point);
                    }
                }
                if (pointpoint == usepoint-1) pstart=pointpoint-53;
                setpointwindow();
                break;
            case RSWITCH:
            case FWD: nxtpoint();
                if (pointpoint > pend-1) {
                    pstart = pointpoint;
                    clear_window(wnd_point);
                }
                if (pointpoint == 0) pstart = 0;
                setpointwindow();
        }
    }
}

```



```

        break;
case PGUP: pstart=pstart-numpts;
        if (pstart < 0) pstart=0;
        pointpoint=pstart+53;
        clear_window(wnd_point);
        setpointwindow();
        break;
case PGDN: pstart = pstart+numpts;
        if (pstart > usepoint+1) pstart=0;
        pointpoint=pstart;
        clear_window(wnd_point);
        setpointwindow();
        break;
case 'V': hide_window(wnd_outhold);
        outholdflag=0;
        viewhold();
        break;
case 'W': display_time(); break;
case F9: setvolume(); break;
case 'T':toggle_infoflag();
        getpoint(usept[pointpoint]);
        showinfopoint();
        break;
case 'M': viewpoint();
        listviewflg=0;
        pstart=pointpoint;
        clear_window(wnd_point);
        setpointwindow();
break;
case DEL: clear_all();
        break;
case ')': zeropoint(usept[pointpoint]);
        writeptread(pointpoint);
        display_points();
        break;
case ENTER: washold=outholdflag;
        timegraph();
        setpointwindow();
        if (outholdflag) outhold();
        else if (washold) delete_window(wnd_outhold);
        helpfunc=help;
        break;
case 'E': vbars();
        setpointwindow();
        break;

```

```

case 'Q': statistics();
           setpointwindow();
           break;
case 'B': pointbars();
           pstart=pointpoint;
           clear_window(wnd_point);
           setpointwindow();
           helpfunc=help;
           break;
case 'O': otherpoint();
           pstart=pointpoint;
           clear_window(wnd_point);
           setpointwindow();
           break;
case 'U': foothand();
           pstart=pointpoint;
           clear_window(wnd_point);
           setpointwindow();
           break;
case CTRL_HOME: pstart=0;
                 pointpoint=0;
                 display_points();
                 break;
case CTRL_END: pstart = usepoint-numpts;
                if (pstart < 0) pstart=0;
                pointpoint=usepoint-1;
                display_points();
                break;
case HOME: pointpoint=pstart;
            display_points();
            break;
case END: pointpoint=pend-1;
            display_points();
            break;
case 'H': if (pointflag) holdbox();
           break;
case 'K': reference(); break;
case '*': if (pointflag) setbase();
           else setpost();
           outhold();
           setpointwindow();
           break;
case 'S': changepoints();
           break;
case 'J': jumptopoint();

```

```

        break;
case ' ': if (meridianflg) {
            meridianflg=0;
            selectpoints();
            pointpoint=savpoint;
        }
        else {
            meridianflg=1;
            meridian();
        }
        clear_window(wnd_point);
        setpointwindow();
        break;
case 'C': close_all();
        showclient();
        pointlist();
        break;
case 'N': notepad(); break;
case 'R': reviewvisits(); break;
case ALT_Z: if (ptarrowcount() != gcount) {
            point[usept[pointpoint]].arrow = '<';
            display_points();
            writeptread(pointpoint);
        }
        else twobells();
        break;
case ALT_X: point[usept[pointpoint]].arrow = ' ';
            display_points();
            writeptread(pointpoint);
            break;
case ALT_Y: clearptarrow();
            display_points();
            break;
case ALT_G: growth();
            break;
case ALT_R:
            if (PORTABLE==KEYPORT)
            {
                putchar(BELL);
                ClearDCM();
                SetupDCM();
                putchar(BELL);
                putchar(BELL);
            }
            break;

```

```

        case F10: if (board) break;
                    if (CARD != 0x4343) break;
                    programpcb();
                    helpfunc=help;
                    showvolts();
                    break;
        case F8: soundpanel();
                    break;
        case 'P': pnum=0;
                    printing(); break;
        case ',': pnum=1;
                    printing(); break;
        default: break;
    }

    }
    setmenuhelp("point ",3,1);
}
outflag=0;
outholdflag=0;
close_all();
}

/***** MOVE POINTER ON SCREEN */

void writemainpoint(long int position)
{
    int fhandle;
    long int poslong;

    if ((fhandle=open("C:\\QA4\\LPOINT\\POINT.1",
        O_WRONLY|O_CREAT|O_BINARY,S_IREAD|S_IWRITE)) == -1) {
        operationinfo("errorop",10,5,iskey);
        exit(1);
    }
    poslong=ptlen*position;
    lseek(fhandle,poslong,SEEK_CUR);
    if (write(fhandle,diskbuffer,ptlen) == -1) {
        operationinfo("errorwt",10,5,iskey);
        exit(1);
    }
    close(fhandle);
}

void writepoint(long int position)
{

```

```

int fhandle;
long int poslong;

if ((fhandle=open("C:\\WORK\\POINT.DAT",
                 O_WRONLY|O_CREAT|O_BINARY,S_IREAD|S_IWRITE)) == -1) {
    operationinfo("errorop",10,5,iskey);
    exit(1);
}
poslong=ptlen*position;
lseek(fhandle,poslong,SEEK_CUR);
if (write(fhandle,diskbuffer,ptlen) == -1) {
    operationinfo("errorwt",10,5,iskey);
    exit(1);
}
close(fhandle);
}

```

```

void movepoint()
{
int key=0;
beep();
while (key != ESC) {
    key=get_char();
    key=toupper(key);
    if (key == ESC) break;
    switch(key) {
        case BS: xpos--; break;
        case FWD: xpos++; break;
        case DN: ypos++; break;
        case UP: ypos--; break;
        case 'D': xpos = xpos-10; break;
        case 'F': xpos = xpos+10; break;
        case 'C': ypos = ypos+10; break;
        case 'R': ypos = ypos-10; break;
        case 'T': xpos=xpos & 0x0FFF; break;
        case 'B': xpos=xpos | 0x9000; break;
        default: break;
    }
    diskbuffer[62] = xpos/256;
    diskbuffer[63] = xpos;
    diskbuffer[64] = ypos/256;
    diskbuffer[65] = ypos;
    writemainpoint(usept[pointpoint]);
    writepoint(usept[pointpoint]);
    displaypointgraph();
}

```

```
    }  
    beep();  
}
```

```
/* START Module for LISTEN Program */  
/* Copyright 1992-9 by James Hoyt Clark */
```

```
#include <time.h>  
#include <stdio.h>  
#include <stdlib.h>  
#include <dos.h>  
#include <graphics.h>  
#include <string.h>  
#include <conio.h>  
#include <math.h>  
#include "twindow.h"  
#include "keys.h"  
#include "vars.h"  
#include "pcl4c.h"
```

```
#define xbar 290
```

```
int nCounter = 0;  
int nCount=0;  
int oldmax = 0;  
int oldmin = 0;  
int oldris = 0;  
int      oldfal = 0;  
unsigned char cPotNum[3];
```

```
extern int nSerFlag;  
extern char msg_line[80];  
//extern unsigned _stklen = 8000;  
extern char name[31];  
extern char id[8];  
extern unsigned char num0;  
extern unsigned char num1;  
extern unsigned char num2;  
extern unsigned char num3;  
extern unsigned char num4;  
extern unsigned char num5;  
extern unsigned char num6;  
extern unsigned char num7;  
extern unsigned char num8;  
extern unsigned char num9;  
extern char biblio;  
extern void (*helpfunc)();  
extern void help();  
extern char findname[31];
```

```

extern void graphmenupoint();
extern int timer0;
extern int wavepoint;
extern float freqvalue;
extern int CARD;
extern int PORTABLE;
extern int SerPort;
extern int TICS;
extern const DELAY;
extern int nBaud;
extern int nDTime;

void MySioPutc(int cValue)
{
if (cValue == FREEZE || cValue == SWITCHOFFAA) cValue--;
SioPutc(SerPort,cValue);
}

// for startup test
int UnFreezeTest(void)
{
int nRead=0;
int i=0;
    cursor(0,10);
    printf("10.Get ");
    SioError(SioGetc(SerPort,TICS));
    nRead = SioGetc(SerPort,TICS);
    cursor(0,11);
    printf("11.960 %#4d ",nRead);
    SioError(SioBaud(SerPort,Baud9600));
    nBaud = Baud9600;
    cursor(0,12);
    printf("12.Get ");
    SioError(SioGetc(SerPort,TICS));
    nRead = SioGetc(SerPort,TICS);
    cursor(0,13);
    printf("13.U1:%#4d Baud:%#3d",nRead, nBaud);
// if (nRead > 4 && nRead < 101) return nRead;
    cursor(0,14);
    printf("14.Put ");
    DoON();
    while (1)
    {
        cursor(0,15);
        printf("15.Get ");

```



```

        SioError(SioGetc(SerPort,TICS));
        nRead = SioGetc(SerPort,TICS);
        printf("Read:%#4d ",nRead);
        if (nRead == 0) break;
        i++;
        if (nRead > 4 && nRead < 101)
        {
            cursor(0,16);
            printf("16.Put%#4d ",nRead);
            return nRead;
        }
        if (nRead == LEFT || nRead == RIGHT) return nRead;
        cursor(0,17);
        if (nRead == FREEZE)
        {
            printf("17.Put%#4d ",nRead);
            DoOFF();
        }
        if (kbhit()) break;
        cursor(0,18);
        printf("18.Wu:%#4d %#4d",i, nRead);
    }
    nRead = SioGetc(SerPort,TICS);
    cursor(0,19);
    printf("19.U2:%#4d",nRead);
    return nRead;
}

```

```

int FreezeTest(void)
{
    int nRead;
    int nIx=0;
    cursor(40,10);
    printf("10.Get ");
    SioError(SioGetc(SerPort,TICS));
    nRead = SioGetc(SerPort,TICS);
    cursor(40,11);
    printf("11.F1:%#4d Baud:%#3d",nRead,nBaud);
    if (nRead) // zero go ahead
    {
        if (nRead == FREEZE) // if already froze, unfreeze
        {
            cursor(40,12);
            printf("12.Put ");
            DoON();
        }
    }
}

```

```

        nRead = 0;
    }
    else return 0; // busy with reading or footpedal
}
cursor(40,13);
printf("13.Put%#4d ",nRead);
DoOFFTest();
while(1) // wait for port to become FREEZE
{
    cursor(40,14);
    printf("14.Get%#4d ",nIx);
    nRead = SioGetc(SerPort,TICS);
    if (nRead < 0) SioError(SioGetc(SerPort,TICS));
    if (nRead == FREEZE)
    {
        cursor(70,14);
        printf("Already");
        break;
    }
    if (nRead > 4 && nRead < 101)
    {
        UnFreezeTest();
        return 0;
    }
    if (nRead == LEFT || nRead == RIGHT)
    {
        UnFreezeTest();
        return 0;
    }
    nIx++;
    if (nIx == 60)
    {
        cursor(40,15);
        nCounter++;
        printf("15.PutON%#7d ",nCounter);
        break;
    }
    cursor(40,16);
    printf("16.Wf:%#7d Read:%#4d FRE:%#4d",nIx,nRead, FREEZE);
    if (kbhit()) break;
}
cursor(40,17);
printf("17.115 ");
SioError(SioBaud(SerPort,Baud115200));
nBaud = Baud115200;

```

```

        cursor(40,18);
        printf("18.F2: %#4d Baud: %#3d", nRead, nBaud);
        return 1;
    }

int UnFreeze(void)
{
    int nRead;
    int i=0;
    DoON();
    while (1)
    {
        nRead = SioGetc(SerPort, TICS);
        if (nRead == 0) break;
        if (nRead > 4 && nRead < 101) return 0;
        if (nRead == LEFT || nRead == RIGHT) return 0;
        i++;
        if (i > 160)
        {
            DoON();
            i=0;
        }
    }
    return nRead;
}

```

```

int Freeze(void)
{
    int nRead;
    int nIx=0;
    nRead = SioGetc(SerPort, TICS);
    if (nRead) // zero go ahead
    {
        if (nRead == FREEZE) UnFreeze();
        return 0;        // busy with reading or footpedal
    }
    DoOFF();
    // nCount++;
    // cursor(60,10);
    // printf("OFF %#5d", nCount);
    while(1) // wait for port to become FREEZE
    {
        nRead = SioGetc(SerPort, TICS);
        if (nRead == FREEZE) break;
        nIx++;
    }
}

```

```

        if (nIx > 80)
        {
            UnFreeze();
            return 0;
        }
    }
    SioBaud(SerPort,Baud115200);
//    cursor(50,10);
//    printf("1152");
    return 1;
}

/* Output */

void MySound(char cFreq)
{
    sound(cFreq*10);
    delay(10);
}

void potencynum(void)
{
    int i;
    unsigned char bPotNum[6];
    pot[dilindex].name[7] = 0;
    for (i=0; i<6; i++)
    {
        if (pot[dilindex].num[i] < 'A')
        {
            if (pot[dilindex].num[i] < '0') bPotNum[i] = 0;
            else bPotNum[i] = pot[dilindex].num[i]-48;
        }
        else bPotNum[i] = pot[dilindex].num[i]-87;
    }
    cPotNum[0] = (bPotNum[0] << 4) | bPotNum[1];
    cPotNum[1] = (bPotNum[2] << 4) | bPotNum[3];
    cPotNum[2] = (bPotNum[4] << 4) | bPotNum[5];
}

void outputcode(char cNumber[3], unsigned char n0, unsigned char n1, unsigned char n2,
    unsigned char n3, unsigned char n4, unsigned char n5, unsigned char n6,
    unsigned char n7, unsigned char n8, unsigned char n9)
{
    int i;
    if (CARD == KEYCARD)

```

```

    { // DCM CARD
    shiftpot(cNumber[0]);
    shiftpot(cNumber[1]);
    shiftpot(cNumber[2]);
    shiftcode(n0);
    shiftcode(n1);
    shiftcode(n2);
    shiftcode(n3);
    shiftcode(n4);
    shiftcode(n5);
    shiftcode(n6);
    shiftcode(n7);
    shiftcode(n8);
    shiftcode(n9);
    if (clicks)
    {
/*      cursor(45,5);
      printf("6>0:%d 1:%d 2:%d",
        cNumber[0],cNumber[1],cNumber[2]);
      cursor(45,6);
      printf(">>0:%d 1:%d 2:%d 3:%d 4:%d 5:%d 6:%d 7:%d 8:%d 9:%d",
        n0,n1,n2,n3,n4,n5,n6,n7,n8,n9);*/
      onvolume();
      xsetpitch(n0);
      xsetpitch(n1);
      xsetpitch(n2);
      xsetpitch(n3);
      xsetpitch(n4);
      xsetpitch(n5);
      xsetpitch(n6);
      xsetpitch(n7);
      xsetpitch(n8);
      xsetpitch(n9);
    }
  }
}

if (PORTABLE == KEYPORT)
{ // DCM PORTABLE
MySioPutc(cNumber[0]);
MySioPutc(cNumber[1]);
MySioPutc(cNumber[2]);
MySioPutc(n0);
MySioPutc(n1);
MySioPutc(n2);
MySioPutc(n3);
MySioPutc(n4);

```

```

        MySioPutc(n5);
        MySioPutc(n6);
        MySioPutc(n7);
        MySioPutc(n8);
        MySioPutc(n9);
        if (clicks)
        {
            MySound(n0);
            MySound(n1);
            MySound(n2);
            MySound(n3);
            MySound(n4);
            MySound(n5);
            MySound(n6);
            MySound(n7);
            MySound(n8);
            MySound(n9);
            nosound();
        }
    }

void out1hold(int j)
{
    if (hold[j].tag) {
        outputcode(hold[j].potnum,hold[j].num0,hold[j].num1,hold[j].num2,hold[j].num3,
            hold[j].num4,hold[j].num5,hold[j].num6,hold[j].num7,
            hold[j].num8,hold[j].num9);
    }
    else offvolume();
}

void xchangeolor()
{
    basecolor=LIGHTRED;
    postcolor=LIGHTBLUE;
    if (pointcolor == RED) pointcolor=LIGHTRED;
    else pointcolor=LIGHTBLUE;
}

void outputhold()
{
    int j;
    if (outholdflag) for (j=0; j<h+2; j++) out1hold(j);
}

```

```

void outputholdPort()
{
int j;
if (outholdflag)
    {
        if (Freeze())
            {
                for (j=0; j<h+2; j++) out1hold(j);
                UnFreeze();
            }
    }
}

```

```

void OutputCapsule()
{
int i=0;
int j;
int nRead;
int nKey;
double dRemainder;
char buffer[2000];
for (j=0; j<h+2; j++)
    {
        if (j>48) break; // limit to 50 dilution and 50 product
        buffer[i] = hold[j].potnum[0];
        i++;
        buffer[i] = hold[j].potnum[1];
        i++;
        buffer[i] = hold[j].potnum[2];
        i++;
        buffer[i] = hold[j].num0;
        i++;
        buffer[i] = hold[j].num1;
        i++;
        buffer[i] = hold[j].num2;
        i++;
        buffer[i] = hold[j].num3;
        i++;
        buffer[i] = hold[j].num4;
        i++;
        buffer[i] = hold[j].num5;
        i++;
        buffer[i] = hold[j].num6;
        i++;
        buffer[i] = hold[j].num7;
    }
}

```

```

        i++;
        buffer[i] = hold[j].num8;
        i++;
        buffer[i] = hold[j].num9;
        i++;
//      cursor(0,10);
//      printf("10. %#4d %#4d",j,i);
    }
while(i<2000) // fill remaining with zeros
    {
        buffer[i] = 0;
        i++;
    }
//cursor(0,11);
//printf("11.Put ");
SioPutc(SerPort,0xAE); // ready for capsule
delay(10);
// output 50 bytes
nRead=0;
for (i=0; i<2000; i++)
    {
//      cursor(0,12);
//      printf("12.Put %#4d %#2d",i,buffer[i]);
        SioPutc(SerPort,buffer[i]);
        dRemainder = fmod(i+1,50); // do batches of 50
        if (!dRemainder && i)
            {
//          cursor(0,13);
//          printf("13.i: %#4d",i);
                while (1)
                {
//          nRead = SioGetc(SerPort,TICS);
//          cursor(0,15);
//          printf("15.Read:%#4d",nRead);
                    if (nRead == 0xAC) break;
                }
                while (1)
                {
//          nRead = SioGetc(SerPort,TICS);
//          cursor(50,23);
//          if (i) printf("Percent:%#4d",(i/20)+1);
                    if (nRead == 0xAD) break;    // end wait
                }
                click();
            }
    }

```



```

    }
}

void outf1(int j)
{
if (filter1[j].tag) {
    outputcode(filter1[j].potnum,filter1[j].num0,filter1[j].num1,filter1[j].num2,
        filter1[j].num3,filter1[j].num4,filter1[j].num5,filter1[j].num6,
        filter1[j].num7,filter1[j].num8,filter1[j].num9);
    }
else offvolume();
}

void outputf1()
{
int j;
if (filter1flag) for (j=0; j<fcnt1+2; j++) outf1(j);
}

void outf2(int j)
{
int mask;
if (filter2[j].tag) {
    outputcode(filter2[j].potnum,filter2[j].num0,filter2[j].num1,filter2[j].num2,
        filter2[j].num3,filter2[j].num4,filter2[j].num5,filter2[j].num6,
        filter2[j].num7,filter2[j].num8,filter2[j].num9);
    }
else offvolume();
}

void outputf2()
{
int j;
if (filter2flag) for (j=0; j<fcnt2+2; j++) outf2(j);
}

void gosetfilter1()
{
wnd_outf1 = establish_window(55,15,3,10);
set_colors(wnd_outf1,ALL,BROWN,BLACK,DIM);
showline("msg_Filter1  ");
set_title(wnd_outf1,msg_line);
}

void outfilter1()

```

```

{
delete_window(wnd_outf1);
gosetfilter1();
countf1tags();
display_window(wnd_outf1);
if (filter1flag) wprintf(wnd_outf1," %#2d  %#2d",fcnt1+1,nf1tags);
else hide_window(wnd_outf1);
}

```

```

void toggle_filter1()
{
if (fcnt1 == -1) return;
else {
    if (filter1flag) filter1flag=0;
    else filter1flag=1;
}
}

```

```

void gosetfilter2()
{
wnd_outf2 = establish_window(55,18,3,10);
set_colors(wnd_outf2,ALL,BROWN,BLACK,DIM);
showline("msg_Filter2  ");
set_title(wnd_outf2,msg_line);
}

```

```

void outfilter2()
{
delete_window(wnd_outf2);
gosetfilter2();
countf2tags();
display_window(wnd_outf2);
if (filter2flag) wprintf(wnd_outf2," %#2d  %#2d",fcnt2+1,nf2tags);
else hide_window(wnd_outf2);
}

```

```

void toggle_filter2()
{
if (fcnt2 == -1) return;
else {
    if (filter2flag) filter2flag=0;
    else filter2flag=1;
}
}

```

```

void zapperserial()
{
WINDOW *wnd_minute;
int saveflag;
int minutes;
clock_t tStart;
saveflag=outholdflag;
outholdflag=1;
wnd_minute = establish_window(3,12,3,21);
set_colors(wnd_minute,ALL,BROWN,AQUA,BRIGHT);
set_colors(wnd_minute,ACCENT,WHITE,BLACK,DIM);
display_window(wnd_minute);
showline("msg_EnterMinutes");
wprintf(wnd_minute,msg_line);
minutes=get_char();
minutes=minutes-0x30;
tStart = clock();
if (minutes > 0 && minutes < 10) {
    wprintf(wnd_minute,"\n %d ",minutes);
    showline("msg_Minutes ");
    wprintf(wnd_minute,msg_line);
    while (!kbhit()) {
        if (PORTABLE == KEYPORT) outputholdPort();
        if (CARD == KEYCARD) outputhold();
        if (clock()-tStart > minutes*1092) break;
    }
}
outholdflag=saveflag;
delete_window(wnd_minute);
beep();
}

void gosetouthold()
{
if (fromkey=='P') {
    wnd_outhold = establish_window(33,23,3,10);
    set_colors(wnd_outhold, ALL, postcolor, WHITE, DIM);
    set_colors(wnd_outhold, ACCENT, WHITE, BLACK, DIM);
}
else {
    wnd_outhold = establish_window(44,15,3,10);
    set_colors(wnd_outhold, ALL, letter1color, WHITE, DIM);
    set_colors(wnd_outhold, ACCENT, accentcolor, BLACK, DIM);
}
showline("msg_HoldTag ");
}

```

```
set_title(wnd_outhold,msg_line);
}
```

```
void outhold()
{
delete_window(wnd_outhold);
gosetouthold();
counttags();
display_window(wnd_outhold);
if (outholdflag) wprintf(wnd_outhold, "%#3d %#3d",h+1,ntags);
else hide_window(wnd_outhold);
}
```

```
void toggle_outhold()
{
if (h == -1) return;
else {
    if (outholdflag) outholdflag=0;
    else outholdflag=1;
}
}
```

```
void holdholddone()
{
++h;
strcpy(hold[h].name,outname);
strcpy(hold[h].id,outid);
strncpy(hold[h].potency,pot[dilindex].name,8);
```

```
hold[h].potnum[0] = cPotNum[0];
hold[h].potnum[1] = cPotNum[1];
hold[h].potnum[2] = cPotNum[2];
hold[h].num0 = outnum0;
hold[h].num1 = outnum1;
hold[h].num2 = outnum2;
hold[h].num3 = outnum3;
hold[h].num4 = outnum4;
hold[h].num5 = outnum5;
hold[h].num6 = outnum6;
hold[h].num7 = outnum7;
hold[h].num8 = outnum8;
hold[h].num9 = outnum9;
hold[h].max = max;
hold[h].min = min;
hold[h].rise = rise;
```

```

hold[h].fall = fall;
hold[h].tag = 1;
writeholdfile();
twobells();
}

```

```

void holdonegraph()
{
if (fromkey == 'I') {
    if (h == nhold) {
        beep();
        return;
    }
    else holdholdone();
}
}

```

```

void holdone()
{
if (outid[0] == ' ') return;
delete_window(wnd_tellhold);
wnd_tellhold = establish_window(41,9,3,36);
showline("msg_PlacedinHold");
set_title(wnd_tellhold,msg_line);
set_colors(wnd_tellhold,ALL,letter3color,WHITE,DIM);
display_window(wnd_tellhold);
if (h == nhold) {
    showline("msg_FULL ");
    wprintf(wnd_tellhold,msg_line);
    twobells();
}
else {
    holdholdone();
    wprintf(wnd_tellhold,"%#d. %s",h+1,outname);
    if (outholdflag) outhold();
}
}

```

```

void holdbox()
{
if (h<0) return;
toggle_outhold();
outhold();
}

```

```

void flbox()
{
    if (fcnt1<0) return;
    toggle_filter1();
    outfilter1();
}

```

```

void f2box()
{
    if (fcnt2<0) return;
    toggle_filter2();
    outfilter2();
}

```

```

void clearholdbox()
{
    setviewport(0,0,639,479,1);
    rectangle(440,452,583,465);
    setviewport(441,453,582,464,1);
    clearviewport();
}

```

```

void outholdgraph()
{
    char string[20];
    clearholdbox();
    counttags();
    setcolor(MAGENTA);
    showline("msg_HoldnumTagnu");
    sprintf(string,msg_line,h+1,ntags);
    if (outholdflag) outtextxy(4,2,string);
}

```

/* INPUT and READING DISPLAY */

```

void display_INPUTtext()
{
    getinput();
    if ((reading > 100) || (reading < 1) || !nSerFlag) return;
    wcursor(wnd_reading,0,0);
    wprintf(wnd_reading,"%#3d",reading);
}

```

```

void clearBAR()
{

```

```

setviewport(301,5,605,420,1);
if (iftgraph) setcolor(BLACK);
else setcolor(bkcolor);
for (by = 15; by < yref; by++) line(xbar,by,xbar+8,by);
}

```

```

void BARandGRAPH()
{
char string[4];
unsigned int biginput;
int i;
getinput();
if ((reading > 100) || (reading < 0) || !nSerFlag) return;
biginput=input*graphscale;
if (iftgraph) setcolor(WHITE);
else setcolor(RED);
setviewport(590,430,639,470,1);
if (biginput != lastinput) clearviewport();
settextstyle(TRIPLEX_FONT,HORIZ_DIR,2);
settextjustify(LEFT_TEXT,TOP_TEXT);
sprintf(string,"%#3d",reading);
outtextxy(0,0,string);
settext1();
setviewport(301,5,605,420,1);
setcolor(lowcolor);
if (biginput >= lastinput) {
    for (by = yref-lastinput; by > yref-biginput; by--) {
        if (by < yref-(readinghi*4.01)) setcolor(normcolor);
        if (by < yref-(readinglow*4.01)) setcolor(hicolor);
        line(xbar,by,xbar+20,by);
    }
}
else {
    setcolor(bkcolor);
    for (by = yref-lastinput; by < yref-biginput; by++)
        line(xbar+9,by,xbar+20,by);
}
if (iftgraph) {
    if (tx0 < 287-rate) {
        if (cleargraphflag) {
            setcolor(bkcolor);
            setviewport(tx1+302,5,tx1+rate+302,420,1);
            clearviewport();
        }
    }
}
}

```

```

        setcolor(addcolor);
        setviewport(301,5,588,420,1);
        ty1 = yref-biginput;
        tx1 = tx1 + rate;
        line(tx0,ty0,tx1,ty1);
        tx0 = tx1;
        ty0 = ty1;
        if (tx0 > 287) {
            setcolor(BLACK);
            if (sillyflag) line(1,0,1,415);
            else clearviewport();
            tx0 = 1;
            tx1 = 1;
            if (!cleargraphflag) {
                addcolor++;
                if (addcolor > 15) addcolor=1;
            }
        }
    }
    lastinput=biginput;
    setviewport(0,0,639,479,1);
    setcolor(pointcolor);
}

void markline(int color)
{
    setcolor(color);
    setviewport(301,5,581,420,1);
    line(tx0,ty0,tx0,ty0-10);
    setviewport(0,0,639,479,1);
}

void INPUT(void)
{
    char getout;
    char string[80];
    clock_t tStart,end;
    long utim,dtim,lev1tim,lev2tim;
    int gowave=0;
    int savewavepoint;
    int count=0;
    char countflag=0;

    getinput();
    if ((reading > 4) && (reading < 101) && nSerFlag) {

```



```

max=0;
min=100;
if (wavepoint) {
    getselectwave();
    gowave=wavepoint;
}
if (gowave) {      /* not DC line */
    if (ifgraphic) clearBAR();
    else clear_window(wnd_reading);
    getout=0;
    while(getout==0) {
        if (ifgraphic) BARandGRAPH();
        else display_INPUTtext();
        if ((reading > 100) || (reading < 1))
        {
            reading = 0;
            break;// return; // look for big number
        }
        if (max < reading) max=reading;
        if (min > reading) min=reading;
        if (reading > (max+min)/2) {
            if (countflag) {
                count++;
                countflag=0;
            }
        }
        if (reading < (max+min)/2) {
            if (countflag == 0) {
                countflag=1;
            }
        }
        if (count > 2) break;
        if (kbhit()) break;
    }
    beep();
    if (iftgraph && ifgraphic) markline(LIGHTBLUE);
    delay(500);
    fall=0;
    rise=0;
    savewavepoint=wavepoint;
    dcline();
    wavepoint=savewavepoint;
}
else {              /* DC line */
    if (pointfindflg) onvolume();
}

```

100;

```
if (ifgraphic) clearBAR();
else clear_window(wnd_reading);
lastreading=0;
getout=0;
lastinput=0;
tStart = clock();
lev2tim=tStart;
lev1tim=tStart;
while(getout==0) {
    if (ifgraphic) BARandGRAPH();
    else display_INPUTtext();
    if ((reading > 100) || (reading < 0)) return; // look for big number reading =

    if (reading < 6) reading = 0;
    if (reading <= lastreading+1) lev1tim=clock();
    else lev2tim=clock();
    if (reading < max || clock()-tStart > timeout*5.0
        || lev1tim-lev2tim > 10) {
        getout=2;
        break;
    }
    if (lastreading-reading > 3) {
        getout=1;
        break;
    }
    lastreading=reading;
    if (max < reading) max=reading;
}
if (getout == 2)
    if (iftgraph && ifgraphic) markline(LIGHTRED);
if (lastreading > reading+10) {
    getout=0;
    lastreading=max;
}
end=clock();
utim=end-tStart;
tStart=clock();
lev2tim=tStart;
lev1tim=tStart;
while(getout>0) {
    if (ifgraphic) BARandGRAPH();
    else display_INPUTtext();
    if ((reading > 100) || (reading < 1))
    {
        reading = 0;
    }
}
```

```

        break;//return; ???
    }
    if (reading < 6) reading = 0;
    if (lastreading-reading > 10) break;
    if (reading >= lastreading) lev1tim=clock();
    else lev2tim=clock();
    if (lev1tim-lev2tim > 75) break;
    dtim=clock();
    if (dtim-tStart > timeout*1.8) break;
    if (reading > max) max=reading;
    lastreading = reading;
}
min=lastreading;
end=clock();
beep();
if (iftgraph && ifgraphic) markline(LIGHTBLUE);
dtim=end-tStart;
onvolume();
if (!max) // < LOWREAD)
{
    offvolume();
    return;
}
if (min == 0)
{
    min=max;
    dtim=0;
}
if (utim == 0) rise = 0;
else rise = abs(max*13/utim);
if (rise > 100) rise=100;
if (dtim == 0) fall=0;
else fall = (max-min)*10/dtim;
if (fall == 0)
    if (max-min) fall = 1;
if (fall > 100) fall=100;
while(reading)
{
    if (kbhit() != 0) break;
    if (ifgraphic) BARandGRAPH();
    else display_INPUTtext();
    if (reading > 100) return; // || (reading < 0)) return;
    if (reading < 5) reading = 0;
    if (max < reading) max=reading;
}

```

```

    }
    if (!max)
    {
        max = oldmax;
        min = oldmin;
        rise = oldris;
        fall = oldfal;
    }
    else
    {
        oldmax = max;
        oldmin = min;
        oldris = rise;
        oldfal = fall;
    }
    showline("msg_MAXMINRISFAL");
    if (ifgraphic) {
        setviewport(301,5,605,420,1);
        if (iftgraph) setcolor(BLACK);
        else setcolor(bkcolor);
        for (by = 15; by < yref+1; by++) line(xbar+8,by,xbar+20,by);
        setcolor(hicolor);
        rectangle(315,467,600,479);
        setviewport(316,468,599,478,1);
        clearviewport();
        if (iftgraph) setcolor(WHITE);
        else setcolor(RED);
        sprintf(string,msg_line,max,min,rise,fall);
        outtextxy(4,2,string);
        setviewport(590,430,639,470,1);
        clearviewport();
        setviewport(0,0,639,479,1);
        ty0 = 420;
    }
    else {
        wprintf(wnd_reading,"\n");
        wprintf(wnd_reading,msg_line,max,min,rise,fall);
    }
    offvolume();
    if (fromkey == 'P') {
        if (pointflag) storepost();
        else storepre();
        if (ifgraphic==1) {
            if (iftgraph == 1) displaypointgraph();
            if (iftgraph == 0) display1pointbar();
        }
    }

```

```

        settex1();
    }
    if (ifgraphic==0) {
        if (fromkey == 'P') display_points();
        else displaypoint();
    }
}
writeptread(pointpoint);
}
}

```

/* Used by Scan & Timegraph */

```

void displaypotgraph()
{
    setcolor(MAGENTA);
    rectangle(253,468,313,479);
    setviewport(254,469,312,478,1);
    clearviewport();
    getdilution();
    outtextxy(4,2,pot[dilindex].name);
    setviewport(0,0,639,479,1);
    moveto(0,0);
}

```

```

void nxtpotgraph()
{
    potpoint++;
    displaypotgraph();
}

```

```

void prevpotgraph()
{
    potpoint--;
    displaypotgraph();
}

```

/* Scan */

```

void clearfilter()
{
    setviewport(1,414,540,423,1);
    clearviewport();
}

```

```

void filter(int j)
{
    char string[65];
    setcolor(letter1color);
    rectangle(0,413,541,424);
    clearfilter();
    sprintf(string,"%#2d. %s %s %#3d %#3d %#3d %#3d %s",j+1,hold[j].name,
    hold[j].id,hold[j].max,hold[j].min,hold[j].rise,hold[j].fall,hold[j].potency);
    outtextxy(4,2,string);
}

```

```

void setchoices(void)
{
    choicestart[choicepoint] = start;
    choiceend[choicepoint] = end;
    choicekey[choicepoint] = key;
    choicestep[choicepoint] = step;
}

```

```

void squares(void)
{
    static int deltax[12] = { 256,128,128,64,64,32,32,16,16,8,8,4 };
    int x,y;
    setcolor(DARKGRAY);
    y=ytop;
    x=0;
    while (y < 388) {
        y = y + x;
        line(xleft,y,xrite,y);
        x=deltay[rowcol-1];
    }
    x=xleft;
    y=0;
    while (x < 532) {
        x = x + y;
        line(x,ytop,x,ybot);
        y=deltax[rowcol-1];
    }
}

```

```

void makeboxes(void)
{
    char keychar[6];
    char string[60];
    char style=1;
}

```

```

int i,k,length;
setviewport(277,5,531,393,1);
clearviewport();
setviewport(0,0,639,479,1);
setfillstyle(style,RED);
bar(x1,y1,x2,y2);
squares();
for (i=0; i<rowcol; i++) {
    if (i == choicepoint) setcolor(RED);
    else setcolor(DARKGRAY);
    if (choicekey[i] == '+') showline("msg_YES      ");
    else showline("msg_NO      ");
    strcpy(keychar,msg_line);
    if (choicekey[i] == 0) {
        length=strlen(msg_line);
        strcpy(keychar," ");
        keychar[length]=0;
    }
    showline("msg_Numbertoscan");
    sprintf(string,msg_line,i+1,keychar,choicestep[i]);
    k=(i+4)*20;
    setviewport(0,k,260,k+10,1);
    clearviewport();
    setviewport(0,0,639,479,1);
    outtextxy(3,k,string);
}
}

```

scan(char withhold)

```

{
char style=1;
long int j,totest,first,differ;
char save_screen[25*80*2];
char string[60];
char string2[60];
char filterpoint=0;
char filterflag=1;
int i,k,m,mask;

```

```

void scanmenu();
helpfunc=scanmenu;
gettext(1,1,80,25,save_screen);
setgraphmode(g_mode);
setbkcolor(bkcolor);
settextstyle(TRIPLEX_FONT, HORIZ_DIR, 3);

```

```

if (withhold) showline("msg_Loading  ");
else showline("msg_Loadingex  ");
outtextxy(50,30,msg_line);
iftgraph=0;
ifgraphic=1;
choicepoint = 0;
totest=0;
if (fromkey == 'I') {
    strcpy(string2,savehead);
    start = itempoint;
    for (j=start; j<nscan+start; j++) {
        getproduct(j);
        if (id[0] == ' ') break;
        if (j > nitem) break;
        scanlist0[totest]=num0;
        scanlist1[totest]=num1;
        scanlist2[totest]=num2;
        scanlist3[totest]=num3;
        scanlist4[totest]=num4;
        scanlist5[totest]=num5;
        scanlist6[totest]=num6;
        scanlist7[totest]=num7;
        scanlist8[totest]=num8;
        scanlist9[totest]=num9;
        scanplace[totest]=j;
        totest++;
        if (withhold == 0) {
            for (i=0; i<h+1; i++) {
                if (strcmp(name,hold[i].name) == 0) totest--;
            }
        }
    }
}
if (fromkey == 'A') {
    strcpy(string2,savehead);
    start = areapoint;
    for (j=start; j<nscan+start; j++) {
        if (j>narea-1) break;
        scanlist0[totest]=area[j].num0;
        scanlist1[totest]=area[j].num1;
        scanlist2[totest]=area[j].num2;
        scanlist3[totest]=area[j].num3;
        scanlist4[totest]=area[j].num4;
        scanlist5[totest]=area[j].num5;
        scanlist6[totest]=area[j].num6;
    }
}

```



```

        scanlist7[totest]=area[j].num7;
        scanlist8[totest]=area[j].num8;
        scanlist9[totest]=area[j].num9;
        scanplace[totest]=j;
        totest++;
    }
}

if (fromkey == '1') {
    showline("msg_Filter1  ");
    strcpy(string2,msg_line);
    start=0;
    for (j=0; j<nscan+start; j++) {
        if (filter1[j].name[0] < 'A') break;
        scanlist0[totest]=filter1[j].num0;
        scanlist1[totest]=filter1[j].num1;
        scanlist2[totest]=filter1[j].num2;
        scanlist3[totest]=filter1[j].num3;
        scanlist4[totest]=filter1[j].num4;
        scanlist5[totest]=filter1[j].num5;
        scanlist6[totest]=filter1[j].num6;
        scanlist7[totest]=filter1[j].num7;
        scanlist8[totest]=filter1[j].num8;
        scanlist9[totest]=filter1[j].num9;
        scanplace[totest]=j;
        totest++;
    }
}

if (fromkey == '2') {
    showline("msg_Filter2  ");
    strcpy(string2,msg_line);
    start=0;
    for (j=0; j<nscan+start; j++) {
        if (filter2[j].name[0] < 'A') break;
        scanlist0[totest]=filter2[j].num0;
        scanlist1[totest]=filter2[j].num1;
        scanlist2[totest]=filter2[j].num2;
        scanlist3[totest]=filter2[j].num3;
        scanlist4[totest]=filter2[j].num4;
        scanlist5[totest]=filter2[j].num5;
        scanlist6[totest]=filter2[j].num6;
        scanlist7[totest]=filter2[j].num7;
        scanlist8[totest]=filter2[j].num8;
        scanlist9[totest]=filter2[j].num9;
        scanplace[totest]=j;
        totest++;
    }
}

```

```

    }
}
if (fromkey == '3') {
    showline("msg_Dilutionof ");
    strcpy(string2,msg_line);
    strcat(string2,outname);
    start=0;
    for (j=0; j<64; j++) {
        scanlist1[totest]=pot[j+1].num[0];
        scanlist2[totest]=pot[j+1].num[1];
        scanlist3[totest]=pot[j+1].num[2];
        scanplace[totest]=j+1;
        totest++;
    }
}
for (j=totest; j<nscan; j++) {
    scanlist1[j]=99; /*set so skip on out*/
    scanlist2[j]=0;
    scanlist3[j]=0;
    scanlist4[j]=0;
    scanlist5[j]=0;
}
for (j=0; j<11; j++) {
    choicestart[j]=0;
    choiceend[j]=0;
    choicekey[j]=0;
    choicestep[j]=0;
}
for (rowcol=1; rowcol<11; rowcol++) {
    step=pow(2,rowcol);
    if (totest==step) break;
    if (totest/step < 1) break;
}
first=start;
end = start+step;
clearviewport();
Bibox(2,2,430);
showline("msg_ScanOperatn ");
outtextxy(15,5,msg_line);
settext1();
showline("msg_Scanning ");
strcpy(string,msg_line);
strcat(string,string2);
outtextxy(150,395,string);
showline("msg_TotalSquares");

```

```

sprintf(string,msg_line,step);
outtextxy(10,45,string);
showline("msg_LoadedSquare");
sprintf(string,msg_line,totest);
outtextxy(10,60,string);
choicestart[choicepoint] = start;
choicend[choicepoint] = end;
choickey[choicepoint] = '+';
choicestep[choicepoint] = step;
differ=step-totest;
x1=xleft;
y1=ytop;
x2=xrite;
y2=ybot;
labelbar();
rectangle(x1,y1,x2,y2);
setfillstyle(style,RED);
bar(x1,y1,x2,y2);
squares();
key=0;
if (outholdflag) outholdgraph();
while (key != ESC)
{
    if (PORTABLE == KEYPORT)
    {
        if (Freeze())
        {
            {
                outputhold();
                for (i=start-first; i<end-first; i++)
                {
                    if (scanlist1[i] == 99) break;
                    outputcode(cPotNum,scanlist0[i],scanlist1[i],scanlist2[i],
                        scanlist3[i],scanlist4[i],scanlist5[i],scanlist6[i],
                        scanlist7[i],scanlist8[i],scanlist9[i]);
                }
                setcolor(bkcolor);
                setviewport(500,420,599,438,1);
                clearviewport();
                setviewport(0,0,639,479,1);
                setcolor(RED);
                UnFreeze();
            }
        }
    }
    if (CARD == KEYCARD)
    {

```

```

    outputhold();
    for (i=start-first; i<end-first; i++)
    {
        if (scanlist1[i] == 99) break;
        outputcode(cPotNum,scanlist0[i],scanlist1[i],scanlist2[i],
            scanlist3[i],scanlist4[i],scanlist5[i],scanlist6[i],
            scanlist7[i],scanlist8[i],scanlist9[i]);
    }
}
setcolor(WHITE);
INPUT();
displayscan=0;
foot=footpedal();
if ((kbhit()) || (foot>0)) {
    if (foot>0) key=foot;
    else {
        key = get_char();
        key = toupper(key);
    }
    switch(key) {
        case LSWITCH: key='-';
        case '-': if (choicepoint == 0) {
                    twobells();
                    break;
                }
                if (scanlist1[end-first] == 99) twobells();
            else {
                start = end;
                end = end+step/2;
                step = end - start;
                setchoices();
                choicepoint++;
                if (y2-y1 == deltay[rowcol-1]) {
                    if (x1 == xleft && x2 == xrite) {
                        y1=y2; /* all line */
                        y2=y1+deltay[rowcol-1];
                        x2=x1+((xrite-xleft)/2);
                    }
                    else {
                        k=x2-x1;
                        x1=x2;
                        x2=x1+k/2;
                    }
                }
            }
        else {

```

```

        k=y2-y1;
        y1=y2;
        y2=y1+((k)/2);
    }
}
displayscan=1;
break;
case '=': key ='+';
case RSWITCH: key='+';
case '+':step = step/2;
        end = start+step;
        step = end - start;
        setchoices();
        choicepoint++;
        if (y2-y1 == deltax[rowcol-1]) x2=((x2-x1)/2)+x1;
        else y2=((y2-y1)/2)+y1;
        displayscan=1;
        break;
case 'H': toggle_outhold();
        outholdgraph();
        break;
case 'F': if (h<0) {
        beep();
        break;
        }
        if (filterflag) {
            filterflag=0;
            filter(filterpoint);
        }
        else {
            filterflag=1;
            clearfilter();
        }
        beep();
        break;
case UP:if (filterflag) break;
        filterpoint--;
        if (filterpoint<0) filterpoint=h;
        filter(filterpoint);
        break;
case DN:if (filterflag) break;
        filterpoint++;
        if (filterpoint>h) filterpoint=0;
        filter(filterpoint);
        break;

```

```

case 'B':if (choicepoint > 1) {
    choicepoint--;
    choicepoint--;
    start = choicestart[choicepoint];
    end = choiceend[choicepoint];
    step = choicestep[choicepoint];
    choicepoint++;
    if (choicekey[choicepoint] == '+') {
        if (x1 == xleft && x2 == xrite)
            y2=((y2-y1)*2)+y1;
        else x2=((x2-x1)*2)+x1;
    }
    else {
        if (y2-y1 == deltax[rowcol-1]) {
            if (x1 == xleft && x2 == xrite) {
                k=y2-y1;    /* up to line */
                y2=y1;
                y1=y1-(k*2);
                displayscan=1;
                beep();
                break;
            }
            if (x1 == xleft) {
                k=y2-y1;    /* up to line */
                y2=y1;
                y1=y1-k;
                x2=x2+(x2-x1);
            }
            else {          /* go left */
                k=x2-x1;
                x2=x1;
                x1=x1-(k*2);
            }
        }
        else {             /* up & double */
            k=y2-y1;
            y2=y1;
            y1=y1-(k*2);
        }
    }
    displayscan=1;
    beep();
}
else beep();
break;

```

```

        case ',': prevpotgraph(); break;
        case '.': nxtpotgraph(); break;
        case 'N': if (fromkey > '4') {
                    start = end-differ;
                    if (fromkey=='A') if (start > narea) start=0;
                    if (fromkey=='I') if (start > nitem-1) start=0;
                    if (start==0) end=0;
                }
                break;
        default: beep(); break;
    }
    if (end == start) break;
    if (key == 'N') break;
    if (displayscan) makeboxes();
    if (filterflag == 0) outlhold(filterpoint);
}

if (fromkey == 'I') start=scanplace[start-first];
backtext();
puttext(1,1,80,25,save_screen);
ifgraphic=0;
iftgraph=1;
if (fromkey > '4') outflag = 1;
return start;
}

/* Timegraph */

void clearoutbox()
{
    setviewport(0,0,639,479,1);
    rectangle(0,468,251,479);
    setviewport(1,469,250,478,1);
    clearviewport();
}

void setlistgraph()
{
    setviewport(0,0,639,479,1);
    moveto(0,0);
    outflag=1;
}

void outareagraph()
{

```

```
setcolor(WHITE);
settextjustify(LEFT_TEXT,TOP_TEXT);
clearoutbox();
outtextxy(4,2,area[areapoint].name);
setlistgraph();
}
```

```
void outitemgraph()
{
getproduct(itempoint);
loadout();
clearoutbox();
setcolor(WHITE);
settextjustify(LEFT_TEXT,TOP_TEXT);
outtextxy(4,2,outname);
setlistgraph();
}
```

```
void upitemgraph()
{
if (itempoint == 0) itempoint = nitem-1;
--itempoint;
outitemgraph();
}
```

```
void downitemgraph()
{
if (itempoint > nitem-3) itempoint = -1;
++itempoint;
outitemgraph();
}
```

```
void upareagraph()
{
if (areapoint == 0) areapoint = narea;
--areapoint;
outareagraph();
}
```

```
void downareagraph()
{
if (areapoint > narea-2) areapoint = -1;
++areapoint;
outareagraph();
}
```



```

void outwhichgraph()
{
    if (fromkey == 'A') outareagraph();
    if (fromkey == 'I') outitemgraph();
}

void tellpointfind()
{
    char string[30];
    if (pointfindflag) {
        setviewport(481,425,579,436,1);
        clearviewport();
    }
    else {
        setcolor(letter1color);
        setviewport(481,425,579,436,1);
        showline("msg_FindPoint ");
        sprintf(string,msg_line);
        outtextxy(4,2,string);
    }
}

void tellexextended()
{
    char string[30];
    if (timeout < 600) {
        setviewport(481,440,600,451,1);
        clearviewport();
    }
    else {
        setcolor(letter1color);
        setviewport(481,440,600,451,1);
        showline("msg_ExtendedTime");
        sprintf(string,msg_line);
        outtextxy(4,2,string);
    }
}

void outputarea(int j)
{
    if (PORTABLE == KEYPORT)
    {
        if (outflag || outholdflag)
        {
            if (Freeze())

```

```

        {
            outputhold();
            if (outflag)
            {

outputcode(cPotNum,area[j].num0,area[j].num1,area[j].num2,area[j].num3,
            area[j].num4,area[j].num5,area[j].num6,area[j].num7,
            area[j].num8,area[j].num9);

            }
            else offvolume();
            UnFreeze();
        }
    }
    return;
}
if (CARD == KEYCARD)
{
    outputhold();
    if (outflag)
    {
        outputcode(cPotNum,area[j].num0,area[j].num1,area[j].num2,area[j].num3,
            area[j].num4,area[j].num5,area[j].num6,area[j].num7,
            area[j].num8,area[j].num9);
    }
    else offvolume();
}
}

void outputitem()
{
    if (PORTABLE == KEYPORT)
    {
        if (outflag || outholdflag || filter1flag || filter2flag)
        {
            if (Freeze())
            {
                outputhold();
                outputf1();
                outputf2();
                if (outflag)
                {
                    outputcode(cPotNum,outnum0,outnum1,outnum2,outnum3,
                        outnum4,outnum5,outnum6,outnum7,outnum8,outnum9);
                }
                else offvolume();
            }
        }
    }
}

```

```

        UnFreeze();
    }
}

if (CARD == KEYCARD)
{
    outputhold();
    outputf1();
    outputf2();
    if (outflag)
    {
        outputcode(cPotNum,outnum0,outnum1,outnum2,outnum3,
            outnum4,outnum5,outnum6,outnum7,outnum8,outnum9);
    }
    else offvolume();
}
}

```

```

void timegraph()
{
    char save_screen[25*80*2];
    extern void (*helpfunc)();
    extern void help();
    int savetimeout;
    int key=0;
    cleargraphflag=1;
    xchange_color();
    savetimeout=timeout;
    if (fromkey == 'P') helpfunc=graphmenupoint;
    else {
        void graphmenulist();
        helpfunc=graphmenulist;
    }
    gettext(1,1,80,25,save_screen);
    setgraphmode(g_mode);
    setbkcolor(BLACK);
    setusercharsize(1,1,1,1);
    settextstyle(SMALL_FONT,VERT_DIR,USER_CHAR_SIZE);
    showline("msg_Amplitude ");
    outtextxy(290,210,msg_line);
    settextstyle(SMALL_FONT,HORIZ_DIR,USER_CHAR_SIZE);
    showline("msg_Time ");
    outtextxy(400,423,msg_line);
    labelbar();
    rectangle(0,4,290,421);
}

```

```

rectangle(300,4,589,421);
settext1();
displaypointgraph();
if (fromkey == 'A') outareagraph();
if (fromkey == 'I') {
    outitemgraph();
    displaypotgraph();
}
tellpointfind();
tellexended();
ifgraphic = 1;
if (outholdflag) outholdgraph();
while (key != ESC) {
    if (fromkey != 'P') {
        if (fromkey == 'A') outputarea(areapoint);
        if (fromkey == 'I') outputitem();
        if (outflag == 0 ) clearoutbox();
    }
    else
    {
        if (PORTABLE == KEYPORT) outputholdPort();
        if (CARD == KEYCARD) outputhold();
    }
    INPUT();
    foot=footpedal();
    if (kbhit() || foot>0) {
        if (foot>0) key=foot;
        else {
            key = get_char();
            key = toupper(key);
        }
        switch(key) {
            case LSWITCH: if (fromkey == 'P') {
                            if (lastanatomy < 91) prevpoint();
                            else nxtpoint();
                            displaypointgraph();
                            break;
                        }
            case UP: if (fromkey == 'A') upareagraph();
                     if (fromkey == 'I') upitemgraph();
                     break;
            case RIGHT: if (fromkey == 'P') {
                            if (lastanatomy < 91) nxtpoint();
                            else prevpoint();
                            displaypointgraph();
                        }
        }
    }
}

```

```

        break;
    }
case DN: if (fromkey == 'A') downareagraph();
        if (fromkey == 'T') downitemgraph();
        break;
case HOME: pointhome();
        break;
case END: pointend();
        break;
case ',': if (fromkey == 'T') prevpotgraph();
        break;
case '.': if (fromkey == 'T') nxtpotgraph();
        break;
case BS: if (lastanatomy < 91) prevpoint();
        else nxtpoint();
        displaypointgraph();
        break;
case FWD: if (lastanatomy < 91) nxtpoint();
        else prevpoint();
        displaypointgraph();
        break;
case ' ': if (fromkey == 'P') outflag=1;
        toggle_outflag();
        if (outflag) outwhichgraph();
        else clearoutbox();
        break;
case 'H': if (pointflag || (fromkey != 'P')) {
        toggle_outhold();
        outholdgraph();
        }
        break;
case '*': if (pointflag) {
        setbase();
        if (outholdflag) clearholdbox();
        }
        else setpost();
        displaypointgraph();
        break;
case 'I': toggle_infotflag();
        showinfograph();
        break;
case 'F': if (pointfindflg) pointfindflg=0;
        else pointfindflg=1;
        twobells();
        tellpointfind();

```

```

        break;
case 'C':toggle_clearflag(); break;
case 'O': otherpoint();
        displaypointgraph();
        break;
case 'U': foothand();
        displaypointgraph();
        break;
case F7: graphmenupoint();
        break;
case 'E': if (timeout != 600) {
        savetimeout = timeout;
        timeout = 600;
        }
        else timeout = savetimeout;
        twobells();
        tellexextended();
        break;
case INS: holdonegraph();
        if (outholdflag) outholdgraph();
        break;
case 'S': if (sillyflag) sillyflag=0;
        else sillyflag=1;
        twobells();
        break;
case ALT_R:
        if (PORTABLE==KEYPORT)
        {
                putchar(BELL);
                ClearDCM();
                SetupDCM();
                putchar(BELL);
                putchar(BELL);
        }
        break;
case '%': movepoint(); break;
default : break;
}

```

```

}
timeout = savetimeout;
ifgraphic=0;
backtext();
puttext(1,1,80,25,save_screen);
lastanatomy=1;

```

```

pointfindflg=1;
basecolor=RED;
postcolor=BLUE;
if (pointcolor == LIGHTRED) pointcolor=RED;
else pointcolor=BLUE;
}

```

```

void setreadwindow()
{
wnd_reading = establish_window(43,22,3,36);
showline("msg_MMRFReading ");
set_title(wnd_reading,msg_line);
set_colors(wnd_reading, ALL, letter2color, WHITE, DIM);
display_window(wnd_reading);
//memory2();
}

```

/* Area & List */

```

void outarea()
{
display_window(wnd_outitem);
if (outflag) wprintf(wnd_outitem,"\\n%s",area[areapoint].name);
else clear_window(wnd_outitem);
}

```

```

void outitem()
{
if (filteroutflag) return;
display_window(wnd_outitem);
if (outflag) wprintf(wnd_outitem,"\\n%s",outname);
else clear_window(wnd_outitem);
}

```

```

void display20item()
{
long int first,j,row;
showline("msg_Items ");
set_title(wnd_item,msg_line);
set_colors(wnd_item,ALL,itemcolor,WHITE,DIM);
set_colors(wnd_item,ACCENT,accentcolor,WHITE,BRIGHT);
display_window(wnd_item);
first = itempoint-9;
row = 0;
if (itempoint < 10 ) {

```

```

        row = 9-itepoint;
        first = 0;
    }
wcursor(wnd_item,0,0);
for (j=0; j<row; j++){
    if (j != 0 ) wprintf(wnd_item, "\n");
    printspace(wnd_item,39);
}
for (j=first; j<itepoint+11; j++) {
    if (row != 0) if ((itepoint+11) != j) wprintf(wnd_item, "\n");
    row = 1;
    getproduct(j);
    if (itepoint==j) {
        reverse_video(wnd_item);
        loadout();
    }
    if (j < nitem-1) wprintf(wnd_item, "%s %s", name, id);
    else printspace(wnd_item,39);
    normal_video(wnd_item);
}
outitem();
}

```

```

void display20area()
{
    unsigned int first,j,row;
    tellareas(subarea,wnd_head);
    showline("msg_Heading ");
    set_title(wnd_area,msg_line);
    display_window(wnd_area);
    first = areapoint-9;
    row = 0;
    if (areapoint < 10 ) {
        row = 9-areapoint;
        first = 0;
    }
wcursor(wnd_area,0,0);
for (j=0; j<row; j++) {
    if (j != 0 ) wprintf(wnd_area, "\n");
    printspace(wnd_area,39);
}
for (j=first; j<areapoint+11; j++) {
    if (row != 0) if ((areapoint+11) != j) wprintf(wnd_area, "\n");
    row = 1;
    if (j < narea) {

```



```

        if (areapoint==j) reverse_video(wnd_area);
        if (area[j].biblio == 'x') wprintf(wnd_area, "  ");
        wprintf(wnd_area, "%s  ", area[j].name);
    }
    else printspace(wnd_area,39);
    normal_video(wnd_area);
}
outarea();
showline("msg_AREALIST  ");
strcpy(savehead,msg_line);
}

```

```

void displaypot(char name[8])
{
    int dilcolor;
    strcpy(potname,name);
    if (dilutionflag) dilcolor=GREEN;
    else dilcolor=BROWN;
    set_colors(wnd_pot,ALL,dilcolor,BLACK,DIM);
    wprintf(wnd_pot, "\n%s",potname);
}

```

```

void getpot(int number)
{
    potpoint=number;
    display_window(wnd_pot);
    getdilution();
    displaypot(pot[dilindex].name);
}

```

```

void setpot()
{
    int len;
    set_colors(wnd_pot,ALL,WHITE,BLACK,DIM);
    for (len=0; len<6; len++) potname[len]=0;
    enterpot();
    /*xxxlen=strlen(findname);
    strncpy(potname,findname,len);*/
    displaypot(potname);
}

```

```

void showhead(void)
{
    display_window(wnd_head);
    wprintf(wnd_head, "\n%s",savehead);
}

```

```
}
```

```
void displayhead(long pointer)
```

```
{
```

```
  getproduct(pointer);
```

```
  if (id[0] == ' ') {
```

```
    display_window(wnd_head);
```

```
    wprintf(wnd_head, "\n%s", name);
```

```
    strcpy(savehead, name);
```

```
    company[4]=id[4];
```

```
    company[5]=id[5];
```

```
    company[6]=id[6];
```

```
    headpoint=itempoint;
```

```
  }
```

```
}
```

```
void displayitem()
```

```
{
```

```
  getproduct(itempoint);
```

```
  displayhead(itempoint);
```

```
  outflag=1;
```

```
  display20item();
```

```
}
```

```
void getarea()
```

```
{
```

```
  long int saveitem;
```

```
  int i=0;
```

```
  saveitem=itempoint;
```

```
  while (area[i].position <= itempoint) i++;
```

```
  i--;
```

```
  itempoint = area[i].position;
```

```
  displayhead(itempoint);
```

```
  itempoint=saveitem;
```

```
}
```

```
void upitem()
```

```
{
```

```
  if (itempoint == 0) itempoint = nitem-1;
```

```
  --itempoint;
```

```
  if (itempoint < headpoint) getarea();
```

```
  displayitem();
```

```
}
```

```
void downitem()
```

```

{
if (itempoint > nitem-3) itempoint = -1;
++itempoint;
displayitem();
}

```

```

void displayarea()
{
outflag=1;
display20area();
}

```

```

void uparea()
{
--areapoint;
if (areapoint < 0) areapoint = narea-1;
displayarea();
}

```

```

void downarea()
{
++areapoint;
if (areapoint > narea-1) areapoint = 0;
displayarea();
}

```

```

void skipdownsubarea()
{
if (subarea > 1) return;
++areapoint;
if (areapoint > narea-1) areapoint = 0;
while (area[areapoint].biblio == 'x') {
    ++areapoint;
    if (areapoint > narea-1) areapoint = 0;
}
displayarea();
}

```

```

void skipupsubarea()
{
if (subarea > 1) return;
--areapoint;
if (areapoint < 0) areapoint = narea-1;
while (area[areapoint].biblio == 'x') {
    --areapoint;
}
}

```

```

        if (areapoint < 0) areapoint = narea-1;
    }
displayarea();
}

void displayheaditem()
{
displayhead(itempoint);
display20item();
}

void downhead()
{
int i;
for (i=0; i<narea; i++) {
    if (strcmp(area[i].name,savehead) == 0) break;
}
i++;
if (i>narea-1) i=0;
itempoint = area[i].position;
displayheaditem();
}

void uphead()
{
int i;
for (i=0; i<narea; i++) {
    if (strcmp(area[i].name,savehead) == 0) break;
}
if (outid[0] == '-') i--;
if (i<0) i=narea-1;
itempoint = area[i].position;
displayheaditem();
}

void skiplines(int number)
{
itempoint = itempoint+number;
if (itempoint > nitem-2) itempoint=nitem-2;
display20item();
}

void setoutwindow()
{
wnd_outitem = establish_window(41,12,3,38);

```

```

showline("msg_Output  ");
set_title(wnd_outitem,msg_line);
set_colors(wnd_outitem,ALL,letter2color,WHITE,BRIGHT);
display_window(wnd_outitem);
}

```

```

void setitemlist()
{
setmenuhelp("item  ",0,0);
readpotfile();
gosetouthold();
gosetfilter1();
gosetfilter2();
wnd_pot = establish_window(30,0,3,10);
showline("msg_Dilution  ");
set_title(wnd_pot,msg_line);
set_colors(wnd_pot,ALL,BROWN,WHITE,DIM);
getpot(potpoint);
wnd_head = establish_window(0,0,3,30);
showline("msg_Heading  ");
set_title(wnd_head,msg_line);
set_colors(wnd_head,ALL,itemcolor,WHITE,DIM);
displayhead(headpoint);
wnd_tellhold = establish_window(41,9,3,30);
showline("msg_PlacedinHold");
set_title(wnd_tellhold,msg_line);
set_colors(wnd_tellhold, ALL, letter3color, WHITE, DIM);
wnd_item = establish_window(0,3,22,40);
gosetptiny();
outflag=1;
setoutwindow();
setreadwindow();
reportflag=0;
outholdflag=0;
filter1flag=0;
filter2flag=0;
}

```

```

void scanitem(char without)
{
if (outid[0] == ' ') downitem();
nextflag=0;
start=scan(without);
itempoint=start;
displayhead(headpoint);
}

```

```

outflag=1;
display20item();
getpot(potpoint);
helpfunc=help;
outhold();
}

```

```

void itemlist()
{
int i;
int c=0;
char footdil=1; /* 1=item, 0=dilutions */
char insleftfoot=1; /* 1=up 0=insert */
char washold;
fromkey='I';
tokey='I';
iftgraph=1;
ifgraphic=0;
setitemlist();
++itempoint;
displayhead(itempoint);
display20item();
while (c != ESC) {
    cursoroff();
    outputitem();
    INPUT();
    foot=footpedal();
    if((kbhit()) || (foot>0)) {
        if (foot>0) c=foot;
        else {
            c = get_char();
            c = toupper(c);
        }
        switch(c) {
            case LSWITCH: if (footdil==0) {
                            prevpot();
                            break;
                        }
                        if (insleftfoot == 0) {
                            holdone();
                            break;
                        }
            case UP: upitem();
                    break;
            case RSWITCH: if (footdil==0) {

```

```

                                nxtpot();
                                }
                                else {
                                    if (reportflag) if (footflag=='F') reportone();
                                    downitem();
                                }
                                break;
case DN: if (reportflag) if (footflag=='K') reportone();
        downitem();
        break;
case '5': skiplines(20); break;
case '6': skiplines(60); break;
case '%': skiplines(-20); break;
case '^': skiplines(-60); break;
case PGDN: downhead(); break;
case PGUP: uphead(); break;
case HOME: itempoint=0;
        displayhead(0);
        outflag=1;
        display20item();
        break;
case END: itempoint = area[narea-1].position;
        displayhead(itempoint);
        itempoint=nitem-2;
        outflag=1;
        display20item();
        break;
case INS: holdone();
        break;
case ALT_I: if (insleftfoot) insleftfoot=0;
        else insleftfoot=1;
        beep();
        break;
case 'P': pnum=0;
        printing(); break;
case ';': pnum=1;
        printing(); break;
case ' ': toggle_outitem(); break;
case 'H': holdbox(); break;
case '1': f1box(); break;
case '2': f2box(); break;
case 'K': reference(); break;
case 'R': toggle_report();
        twobells();
        break;

```

```

case 'V': hide_window(wnd_tellhold);
           hide_window(wnd_outhold);
           outholdflag=0;
           viewhold();
           break;
case '!': hide_window(wnd_outf1);
           filter1flag=0;
           viewfilter1();
           break;
case '@': hide_window(wnd_outf2);
           filter2flag=0;
           viewfilter2();
           break;
case '$': timeinfo(); break;
case 'L': showreport(); break;
case 'Q': findq();
           getarea();
           break;
case 'F': finditem();
           getarea();
           break;
case ALT_F: findnum();
           getarea();
           break;
case 'W': display_time(); break;
case F9: if (CARD == 0x4545) setvolume(); break;
case F10: areapoint=selectarea();
           itempoint = area[areapoint].position;
           displayheaditem();
           break;
case F8: areapoint=selectAZarea();
           itempoint = area[areapoint].position;
           displayheaditem();
           break;
case ENTER: washold=outholdflag;
            timegraph();
            outflag=1;
            display20item();
            displaypot(potname);
            if (outholdflag) outhold();
            else if (washold) delete_window(wnd_outhold);
            helpfunc=help;
            displaypoint();
            break;
case 'S': setscan(); break;

```



```

case ',': prevpot(); break;
case '.': nxtpot(); break;
case ALT_Z: scanitem(0);
            break;
case 'Z': scanitem(1);
            break;
case '+': plusstorepost(); break;
case 'M': toggle_dilution(); break;
case 'A': potpoint=0;
            getpot(0);
            break;
case ALT_D: if (footdil) footdil=0;
            else footdil=1;
            beep();
            break;
case 'D': alldilutions();
            helpfunc=help;
            break;
case ALT_X: switchdilution('X'); break;
case ALT_C: switchdilution('C'); break;
case ALT_M: switchdilution('M'); break;
case 'C': close_all();
            showclient();
            setitemlist();
            display20item();
            break;
case 'N': notepad(); break;
case 'E': vbars(); break;
case BS: prevpoint();
            displaypoint();
            break;
case FWD: nxtpoint();
            displaypoint();
            break;
case 'O': otherpoint();
            displaypoint();
            break;
case 'U': foothand();
            displaypoint();
            break;
case 'J': jumptopoint(); break;
case '*': if (pointflag) setbase();
            else setpost();
            if (h>-1) outhold();
            displaypoint();

```

```

                break;
        case '~': if (h > -1) {
                        zapperserial();
                        break;
                }
        case '#': OutputCapsule();
                        click();
                        click();
                        break;
        case F2: if (company[6] != ' ') {
                        productinfo();
                }
                else beep();
                break;
        case ESC: break;
        default: break;
    }
    }
    setmenuhelp("item ",0,0);
}

offvolume();
outflag=0;
reportflag=0;
outholdflag=0;
tokey='I';
close_all();
}

void itemlistbegin(void)
{
    itempoint=0;
    itemlist();
}

void setarealist()
{
    iftgraph=1;
    ifgraphic=0;
    fromkey='A';
    tokey='A';
    setmenuhelp("area ",0,0);
    wnd_head = establish_window(8,0,3,26);
    set_colors(wnd_head,ALL,areacolor,WHITE,DIM);
    wnd_area = establish_window(0,3,22,40);
    set_colors(wnd_area,ALL,areacolor,WHITE,DIM);
}

```

```

set_colors(wnd_area,ACCENT,accentcolor,WHITE,BRIGHT);
gosetouthold();
gosetptiny();
setoutwindow();
setreadwindow();
displayarea();
}

```

```

void readareafile()
{
char filename[10];
FILE *fptr;
narea=0;
strcpy(filename,"AREA1.DAT");
if( (fptr=fopen(filename,"rb"))==NULL ) {
    operationinfo("errorfl",10,5,yeskey);
    exit(0);
}
else {
    while( fread(&area[narea],sizeof(area[0]),1,fptr)==1 ) narea++;
    fclose(fptr);
    gotoxy(1,21);
}
}

```

```

void writeareafile()
{
FILE *fptr;
if( (fptr=fopen("AREA1.DAT","wb"))==NULL ) {
    operationinfo("errorfl",10,5,yeskey);
}
else fwrite(area,sizeof(area[0]),narea,fptr);
fclose(fptr);
}

```

```

void arealist(void)
{
int c=0;
//int nCount5=0;
//int nCount6=0;
areapoint=0;
reportflag=0;
setarealist();
while (c != ESC) {
    cursoroff();
}

```

```

        outputarea(areapoint);
//      cursor(26,9);
//      nCount5++;
//      printf("Count5=%d",nCount5);
      INPUT();
//      cursor(26,10);
//      nCount6++;
//      printf("Count6=%d",nCount6);
      foot=footpedal();
      if((kbhit()) || (foot > 0)) {
          if (foot>0) c=foot;
          else {
              c = get_char();
              c = toupper(c);
          }
          switch(c) {
              case LSWITCH:
              case UP: uparea(); break;
              case RSWITCH:
              case DN: downarea(); break;
              case ' ': toggle_outarea(); break;
              case 'H': holdbox(); break;
              case 'W': display_time(); break;
              case F9: if (CARD == 0x4343) setvolume(); break;
              case ENTER: timegraph();
                          display20area();
                          outholdflag=0;
                          helpfunc=help;
                          displaypoint();
                          break;
              case 'V': hide_window(wnd_outhold);
                          outholdflag=0;
                          viewhold();
                          break;
              case 'L': showreport(); break;
              case HOME: areapoint=0;
                          display20area();
                          break;
              case END: areapoint=narea-1;
                          display20area();
                          break;
              case F10: areapoint=selectarea();
                          display20area();
                          break;
              case F8: areapoint=selectAZarea();

```

```

        display20area();
        break;
case 'K': reference(); break;
case 'G': close_all();
        itempoint = area[areapoint].position;
        headpoint=itempoint;
        itemlist();
        setarealist();
        break;
case 'S': setscan(); break;
case '$': timeinfo(); break;
case 'Z': areapoint=scan(1);
        display20area();
        helpfunc=help;
        break;
case 'O': otherpoint();
        displaypoint();
        break;
case 'U': foothand();
        displaypoint();
        break;
case 'J': jumptopoint(); break;
case '+': plusstorepost(); break;
case 'P': pnum=0;
        printing(); break;
case ';': pnum=1;
        printing(); break;
case 'C': close_all();
        showclient();
        setarealist();
        displayarea();
        break;
case 'N': notepad(); break;
case 'E': vbars(); break;
case BS: prevpoint();
        displaypoint();
        break;
case FWD: nxtpoint();
        displaypoint();
        break;
case '*': if (pointflag) setbase();
        else setpost();
        displaypoint();
        break;
case PGDN: skipdownsubarea(); break;

```

```

        case PGUP: skipupsubarea(); break;
        case '!': if (clicks)
            {
                offvolume();
                clicks=0;
            }
            else clicks=1;
            putchar(BELL);
            break;
        case ESC: break;
        default: break;
    }
    setmenuhelp("area ",0,0);
}
offvolume();
outflag=0;
close_all();
tokey='A';
}

```

```
/* UTILITY Module for LISTEN Program */
/* Copyright 1992-8 by James Hoyt Clark */
```

```
#include <dos.h>
#include <conio.h>
#include <stdarg.h>
#include <stdio.h>
#include <stdlib.h>
#include <graphics.h>
#include <bios.h>
#include <alloc.h>
#include <sys\types.h>
#include <sys\timeb.h>
#include <sys\stat.h>
#include <fcntl.h>
#include <time.h>
#include <string.h>
#include "twindow.h"
#include "keys.h"
#include "pcl4c.h"
```

```
#define COM 0      /* serial port number */
#define CONF 0xE3 /* 9600 baud, no parity, 1 stop, 8 bits */
#define TEXTCOLOR WHITE
#define PANELBACK DARKGRAY
#define WAVEBKCOLOR BLUE
#define LINECOLOR WHITE
void beep(void); //?????????
/*
01234567890123456789012345678901234567890123456789012345678901234567
89012345678901234567890*/
char code[]="COPYRIGHT (c) 1999 by James Hoyt Clark\n\n  written and developed by James
Hoyt Clark      ";
```

```
extern const TICS;
extern int nTimes;
extern int nGetBuffer;
extern int nPortDelay;
int nFootDelay;
```

```
/* General Variables */
extern struct textsettingstype oldtext;
extern int g_driver;
extern int g_mode;
extern int g_error;
```

```
extern unsigned int nscanum;
extern int potpoint;
extern int npot;
extern int foot; //?????
extern char displayscan;
extern char infoflag;
extern char cleargraphflag;
extern char dilutionflag;
extern char outflag;
extern int pointcolor;
extern int letter3color;
extern char pointflag;
extern char outholdflag;
extern char ifgraphic;
extern char iftgraph;
extern char pnum;
extern char userptcode;
```

```
extern unsigned char waveamplitude[256];
extern unsigned char plotamp[256];
extern unsigned char voltage;
extern unsigned char gain;
extern unsigned char max;
extern unsigned char min;
extern unsigned char rise;
extern unsigned char fall;
extern int wavepoint;
extern int timer0; /*frequency*/
extern int timer1;
extern int timer2;
extern int freqcount;
extern int freqpoint;
extern int freqsave;
```

```
extern char msg_line[80];
extern char octave;
extern FILE *linefp;
```

```
/*Defaults*/
extern char savamp;
extern char amp;
extern unsigned int nscan;
extern int timeout;
extern int rate;
extern unsigned char readinglow,readinghi;
```


extern int reading;

extern int letter1color;
extern int letter2color;
extern int hicolor;
extern int normcolor;
extern int lowcolor;
extern int bkcolor;
extern int accentcolor;
extern int basecolor;
extern int postcolor;
extern int areacolor;
extern int itemcolor;
extern int whichallergy;
extern int clientdrive;
extern char footflag;
extern char language;
extern unsigned int nitem;
extern int narea;
extern long int npoint;
extern long int usepoint;
extern int usedeletes;
extern int useprints;
extern int useA;
extern int useB;
extern int vtype1;
extern int vtype2;
extern int vtype3;
extern int vtype4;
extern int amount;

extern char multi_am;
extern char multi_vib;
extern char multi_egtyp;
extern char multi_ksr;
extern char multi;

extern char tone_ksl;
extern char tone_level;
extern char tone_dc;
extern char tone_dm;
extern char tone_fb;

extern char attack;
extern char decay;

```
extern char suslevel;  
extern char relrate;
```

```
extern char rhy_mel;  
extern char drum_BD;  
extern char drum_SD;  
extern char drum_TOM;  
extern char drum_TCY;  
extern char drum_HH;
```

```
extern unsigned char pitch;
```

```
extern char susON_OFF;  
extern char keyON_OFF;  
extern char octave;  
extern char f_num8;
```

```
extern char instrument;  
extern char volume;
```

```
extern char percussion;
```

```
extern char bd_vol;  
extern char hh_vol;  
extern char sd_vol;  
extern char tom_vol;  
extern char tcy_vol;
```

```
extern int CARD;  
extern int PORTABLE;
```

```
extern struct potrecord  
{  
    char name[8];  
    char num[6];  
    char CR;  
    char LF;  
};  
extern struct potrecord pot[1000];
```

```
extern struct reportrecord  
{  
    char name[31];  
    char id[8];  
    unsigned char potency[11];
```

```

        unsigned char max;
        unsigned char min;
        unsigned char rise;
        unsigned char fall;
    };
extern struct reportrecord report[170];

extern int nreport;    /* Item Readings capacity */
extern int reportpoint; /* Record pointer */
extern char reportflag;
extern char outname[31];
extern char outid[8];
extern int dilindex;
void(*helpfunc)();
extern void help();
int helpkey=F1;
int helping=0;
int totalvisit;
double totaltime;
char govern;
char restoreflag;
char operatorname[31];
int pageline;
int SerPort;
int OUTPUTCOUNT;

char *Fonts[] = {
    "DefaultFont","TriplexFont","SmallFont","SansSerifFont","GothicFont"
};

char *Language[] = {
    "ENGLISH ","ESPANOL ","DEUTSCH ","FRANCAIS"
};

char *Colors[] = {
    "BLACK ","BLUE  ","GREEN  ","CYAN  ","RED    ","MAGENTA  ",
    "BROWN  ","LIGHTGRAY ","DARKGRAY ","LIGHTBLUE
", "LIGHTGREEN","LIGHTCYAN ",
    "LIGHTRED ","LIGHTMAGEN","YELLOW  ","WHITE   "
};

static int EGAnum[16] = {0,1,2,3,4,5,20,7,56,57,58,59,60,61,62,63};

char *LineStyle[] = {
    "SolidLn", "DottedLn", "CenterLn", "DashedLn", "UserBitLn"

```

```
};
```

```
char *FillStyles[] = {  
    "EmptyFill", "SolidFill",    "LineFill",    "LtSlashFill",  
    "SlashFill", "BkSlashFill",  "LtBkSlashFill", "HatchFill",  
    "XHatchFill", "InterleaveFill", "WideDotFill", "CloseDotFill"  
};
```

```
char *TextDirect[] = {  
    "HorizDir", "VertDir"  
};
```

```
int  GraphDriver; /* The Graphics device driver */  
int  GraphMode;   /* The Graphics mode value   */  
int  MaxX, MaxY;  /* The maximum resolution of the screen */  
int  MaxColors;   /* The maximum # of colors available */  
unsigned char printerstatus;  
unsigned char prtmsg;  
int defaultvalues[100];  
WINDOW *wnd_timing;  
WINDOW *wnd_report;  
double tused, tperstep;  
time_t tstart, tstop;  
struct timeb time_buffer;
```

```
void xxxbells()  
{  
}
```

```
void twobells()  
{  
    putchar(BELL);  
    putchar(BELL);  
}
```

```
int set_ctl_brk(int which)  
{  
    union REGS ireg;  
    ireg.h.ah = 0x33;  
    ireg.h.al = 0x01;  
    ireg.h.dl = which;  
    intdos(&ireg, &ireg);  
    return ireg.h.dl;  
}
```

```

int get_char()
{
int c;
static union REGS rg;
offvolume();
//set_ctl_brk(0);
while (1) {
    rg.h.ah = 1;
    int86(0x16,&rg,&rg);
    if (rg.x.flags & 0x40) {
        int86(0x28,&rg,&rg);
        continue;
    }
    rg.h.ah = 0;
    int86(0x16,&rg,&rg);
    if (rg.h.al == 0) c = rg.h.ah | 128;
    else c = rg.h.al;
    if (c == helpkey && helpfunc) {
        if (!helping) {
            helping = 1;
            (*helpfunc)();
            helping = 0;
            return 0xff;
        }
    }
    // if (c == FREEZE)
    // {
    //     UnFreeze();
    //     nTimes++;
    //     printf("Tc: %#4d",nTimes);
    //     c=0;
    // }
    break;
}
return c;
}

void settext1()
{
setusercharsize(1,1,1,1);
settextjustify(LEFT_TEXT,TOP_TEXT);
settextstyle(0,HORIZ_DIR,USER_CHAR_SIZE);
}

void settextc()

```

```

{
settextjustify( CENTER_TEXT, TOP_TEXT );
settextstyle(DEFAULT_FONT,HORIZ_DIR,USER_CHAR_SIZE);
}

```

```

void Bibox(int size,int x,int y)
{
setfillstyle(SOLID_FILL,LIGHTGRAY);
bar(x+1,y+1,x+44,y+44);
setcolor(BLUE);
setlinestyle(SOLID_LINE,0,THICK_WIDTH);
rectangle(x+6,y+6,x+39,y+39);
setlinestyle(SOLID_LINE,0,NORM_WIDTH);
rectangle(x+3,y+3,x+42,y+42);
rectangle(x,y,x+45,y+45);
settextstyle(TRIPLEX_FONT,HORIZ_DIR,size);
outtextxy(x+15,y+8,"Bi");
outtextxy(x+50,y-2,"BioSource");
}

```

```

void cursoroff()
{
union REGS regs;
regs.h.ch=0x20;
regs.h.ah=1;
int86(0x10,&regs,&regs);
}

```

```

void cursoron()
{
union REGS regs;
regs.h.ch=0x0c;
regs.h.cl=0x0d;
regs.h.ah=1;
int86(0x10,&regs,&regs);
}

```

```

void backtext()
{
restorecrtmode();
cursoroff();
}

```

```

void makesound(int freq, unsigned time)
{

```

```

if (CARD == KEYCARD) setpitch(freq,5,11);
onvolume();
delay(time);
offvolume();
}

void makesoundPC(int freq, unsigned time)
{
sound(freq);
delay(time);
nosound();
}

void click()
{
if (CARD == KEYCARD) makesound(10,30);
if (PORTABLE == KEYPORT) makesoundPC(500,5);
}

void beep()
{
if (CARD == KEYCARD) makesound(100,100);
if (PORTABLE == KEYPORT) makesoundPC(1000,10);
}

void bleep()
{
if (CARD == KEYCARD) makesound(150,50);
if (PORTABLE == KEYPORT) makesound(2000,15);
}

void changetextstyle(int font, int direction, int charsize)
{
int ErrorCode;

graphresult();      /* clear error code */
settextstyle(font, direction, charsize);
ErrorCode = graphresult(); /* check result */
if( ErrorCode != grOk ){ /* if error occurred */
    closegraph();
    printf("Graphics System Error: %s\n", grapherrormsg( ErrorCode ) );
    exit(1);
}
}

```

```

int gprintf( int *xloc, int *yloc, char *fmt, ... )
{
    va_list argptr;    /* Argument list pointer */
    char str[140];      /* Buffer to build string into */
    int cnt;            /* Result of SPRINTF for return */

    va_start( argptr, fmt ); /* Initialize va_ functions */
    cnt = vsprintf( str, fmt, argptr ); /* prints string to buffer */
    outtextxy( *xloc, *yloc, str ); /* Send string in graphics mode */
    *yloc += textheight( "H" ) + 2; /* Advance to next line */

    va_end( argptr ); /* Close va_ functions */
    return( cnt ); /* Return the conversion count */
}

```

```

void gprintline( int *xloc, int *yloc, char *line)
{
    va_list argptr;    /* Argument list pointer */
    char str[140];      /* Buffer to build string into */

    va_start( argptr, line ); /* Initialize va_ functions */
    vsprintf( str, line, argptr ); /* prints string to buffer */
    outtextxy( *xloc, *yloc, str ); /* Send string in graphics mode */
    *yloc += textheight( "H" ) + 2; /* Advance to next line */

    va_end( argptr ); /* Close va_ functions */
}

```

```

void TopLine(char *msg)
{
    int height;
    char string[80];
    struct viewporttype vp;
    setlinestyle(SOLID_LINE,0,NORM_WIDTH);
    setttextjustify(CENTER_TEXT,TOP_TEXT);
    height = textheight("H");
    getviewsettings(&vp);
    setfillstyle(SOLID_FILL,DARKGRAY);
    bar(2,2,vp.right-vp.left-2,height+8);
    setcolor(MaxColors-1); /*white*/
    rectangle(0,0,vp.right-vp.left,height+8);
    outtextxy((vp.right-vp.left)/2,6,msg);
}

```

```

void BottomLine(char *msg)

```



```

{
int height;
struct viewporttype vp;
setlinestyle(SOLID_LINE,0,NORM_WIDTH);
settextjustify(CENTER_TEXT,TOP_TEXT);
height = textheight("H");
getviewsettings(&vp);
setfillstyle(SOLID_FILL,DARKGRAY);
bar(2,vp.bottom-(vp.top+height+4),vp.right-vp.left-2,vp.bottom-vp.top-2);
setcolor(MaxColors-1); /*white*/
rectangle(0,vp.bottom-(vp.top+height+4),vp.right-vp.left,vp.bottom-vp.top);
outtextxy((vp.right-vp.left)/2,vp.bottom-(vp.top+height+2),msg);
}

```

```

void DrawBorder(void)
{
struct viewporttype vp;
setcolor(MaxColors-1); /*white*/
setlinestyle(SOLID_LINE,0,THICK_WIDTH);
getviewsettings(&vp);
rectangle(0,0,vp.right-vp.left,vp.bottom-vp.top);
setfillstyle(SOLID_FILL,LIGHTCYAN);
bar(2,2,vp.right-vp.left-2,vp.bottom-vp.top-2);
setlinestyle( SOLID_LINE, 0, NORM_WIDTH );
}

```

```

void graphmenuwindow(void)
{
setviewport(306,25,580,370,1);
DrawBorder();
}

```

```

void graphmenupoint(void)
{
graphmenuwindow();
operationgraph("PtTMenu",5,30,yeskey);
}

```

```

void graphmenulist(void)
{
graphmenuwindow();
operationgraph("LtTMenu",5,30,yeskey);
}

```

```

void scanmenu(void)

```

```

{
graphmenuwindow();
operationgraph("ScnMenu",5,30,yeskey);
makeboxes();
}

```

```

void barsmenu(void)
{
graphmenuwindow();
operationgraph("PtBarMn",5,22,yeskey);
display_pointbars();
}

```

```

void panelmenu(void)
{
setviewport(2,2,518,261,1);
DrawBorder();
settextstyle(SMALL_FONT,HORIZ_DIR,USER_CHAR_SIZE);
operationgraph("PanelMn",5,22,yeskey);
setviewport(0,0,639,479,1);
settextjustify(LEFT_TEXT,TOP_TEXT);
whichwave();
}

```

```

void information()
{
int key = 0;
WINDOW *wnd_info;
close_all();
wnd_info=establish_window(6,10,9,60);
set_colors(wnd_info,ALL,BROWN,BLACK,DIM);
display_window(wnd_info);
wprompt(wnd_info,4,1,code);
wcursor(wnd_info,4,5);
showline("msg_CustomL  ");
wprintf(wnd_info,"%s %s",msg_line,operatorname);
cursoroff();
while (key != ESC)  key = get_char();
delete_window(wnd_info);
}

```

```

void setscan()
{
WINDOW *wnd_scan;
int key=0;

```

```

wnd_scan = establish_window(74,0,3,6);
showline("msg_Scan ");
set_title(wnd_scan,msg_line);
set_colors(wnd_scan,ALL,BROWN,WHITE,DIM);
display_window(wnd_scan);
while (key != ESC) {
    wprintf(wnd_scan, "\n%#4d",nscan);
    key = get_char();
    key = toupper(key);
    switch (key) {
        case UP: nscan = nscan*2;
                 nscanum++;
                 if (nscan > 1024) {
                     nscan = 4;
                     nscanum = 2;
                 }
                 break;
        case DN: nscan = nscan/2;
                 nscanum--;
                 if (nscan < 2 ) {
                     nscan = 1024;
                     nscanum = 10;
                 }
                 break;
        default: break;
    }
}
delete_window(wnd_scan);
}

void start_time()
{
    ftime(&time_buffer);
    time(&tstart);
}

void stop_time()
{
    time(&tstop);
}

void display_time()
{
    struct tm *tm_now;
    long secs_now;

```

```

long past_secs=0;
char *str_now;
WINDOW *wnd_time;
wnd_time = establish_window(5,10,3,28);
set_colors(wnd_time, ALL, MAGENTA, WHITE, DIM);
display_window(wnd_time);
while (kbhit() == 0) {
    cursoroff();
    time(&secs_now);
    tm_now = localtime(&secs_now);
    str_now = asctime(tm_now);
    str_now[24] = 0;
    if (secs_now != past_secs) {
        wcursor(wnd_time,0,0);
        wprintf (wnd_time, "\n %s", str_now);
        past_secs=secs_now;
    }
}
get_char();
delete_window(wnd_time);
}

void nxtpot()
{
    potpoint++;
    getpot(potpoint);
}

void prevpot()
{
    potpoint--;
    getpot(potpoint);
}

void clickwaitclear(void)
{
    click();
    outportb(PPICTLREG,0x82); /* A=OUT */
    delay(150);
    outportb(PPIPORTA,0xFF);
    delay(150);
    outportb(PPICTLREG,0x92); /* A=IN, B=IN, C=OUT */
    delay(50);
}

```

```

int footpedal(void)
{
int nFoot;
if (CARD == KEYCARD)
{
nFoot=inportb(PPIPORTA) & 0x03;
if (nFoot == 0x03) nFoot=0;
if (nFoot == LFOOT)
{
clickwaitclear();
nFoot = LSWITCH;
}
if (nFoot == RFOOT)
{
clickwaitclear();
nFoot = RSWITCH;
}
}
if (PORTABLE == KEYPORT)
{
nFoot = GetPortable();
if (nFoot == RIGHT || nFoot == LEFT)
{
if (nFoot == LEFT) nFoot = LSWITCH;
if (nFoot == RIGHT) nFoot = RSWITCH;
click();
delay(nFootDelay);
SioRxFlush(SerPort);
}
else nFoot=0;
}
return nFoot;
}

```

```

void printspace(WINDOW *wnd,int count)
{
int i;
for (i=0; i<count; i++) wprintf(wnd," ");
}

```

```

void labelbar()
{
char temp[3];
int j;
for (j=0; j<11; j++) {

```

```

        itoa(j*10,temp,10);
        outtextxy(615,417-j*40,temp);
        line(608,420-j*40,613,420-j*40);
    }
}

```

```

void barheading(int which)
{
    char str[80];
    printf();
    showline("msg_BarGraph  ");
    strcpy(str,msg_line);
    toprinter(str);
    printf('N');
    if (which) showline("msg_Post  ");
    else showline("msg_Base  ");
    strcpy(str,msg_line);
    showline("msg_MMRF050100 ");
    strcat(str,msg_line);
    toprinter(str);
    printf();
}

```

```

void toggle_infocflag()
{
    if (infocflag) infocflag=0;
    else infocflag=1;
    putchar(BELL);
}

```

```

void plusstorepost()
{
    storepost();
    displaypoint();
    twobells();
}

```

```

void setvolume()
{
    int key=0;
    WINDOW *wnd_scan;
    wnd_scan = establish_window(72,0,3,8);
    showline("msg_Volume  ");
    set_title(wnd_scan,msg_line);
    set_colors(wnd_scan, ALL, MAGENTA, WHITE, DIM);
}

```

```

setmenuhelp("volume ",0,0);
display_window(wnd_scan);
while (key != ESC) {
    wprintf(wnd_scan, "\n  %#2d", savamp);
    key = get_char();
    key = toupper(key);
    switch(key) {
        case LSWITCH:
        case UP: savamp++;
                if (savamp > 31) savamp = 0;
                beep();
                break;
        case RSWITCH:
        case DN: savamp--;
                if (savamp < 0 ) savamp = 31;
                beep();
                break;
        default: putchar(BELL); break;
    }
}
delete_window(wnd_scan);
}

```

```

void toggle_cleargflag()
{
    if (cleargraphflag) cleargraphflag = 0;
    else cleargraphflag = 1;
    putchar(BELL);
}

```

```

void toggle_dilution()
{
    if (dilutionflag) dilutionflag = 0;
    else dilutionflag = 1;
    putchar(BELL);
    getpot(potpoint);
}

```

```

void toggle_outflag()
{
    if (outflag) outflag=0;
    else outflag=1;
}

```

```

void toggle_outarea()

```

```
{
toggle_outflag();
outarea();
}
```

```
void toggle_outitem()
{
toggle_outflag();
outitem();
}
```

```
void getoperator()
{
operatorname[0]=defaultvalues[85]/256;
operatorname[1]=defaultvalues[85] & 0xff;
operatorname[2]=defaultvalues[86]/256;
operatorname[3]=defaultvalues[86] & 0xff;
operatorname[4]=defaultvalues[87]/256;
operatorname[5]=defaultvalues[87] & 0xff;
operatorname[6]=defaultvalues[88]/256;
operatorname[7]=defaultvalues[88] & 0xff;
operatorname[8]=defaultvalues[89]/256;
operatorname[9]=defaultvalues[89] & 0xff;
operatorname[10]=defaultvalues[90]/256;
operatorname[11]=defaultvalues[90] & 0xff;
operatorname[12]=defaultvalues[91]/256;
operatorname[13]=defaultvalues[91] & 0xff;
operatorname[14]=defaultvalues[92]/256;
operatorname[15]=defaultvalues[92] & 0xff;
operatorname[16]=defaultvalues[93]/256;
operatorname[17]=defaultvalues[93] & 0xff;
operatorname[18]=defaultvalues[94]/256;
operatorname[19]=defaultvalues[94] & 0xff;
operatorname[20]=defaultvalues[95]/256;
operatorname[21]=defaultvalues[95] & 0xff;
operatorname[22]=defaultvalues[96]/256;
operatorname[23]=defaultvalues[96] & 0xff;
operatorname[24]=defaultvalues[97]/256;
operatorname[25]=defaultvalues[97] & 0xff;
operatorname[26]=defaultvalues[98]/256;
operatorname[27]=defaultvalues[98] & 0xff;
operatorname[28]=defaultvalues[99]/256;
operatorname[29]=defaultvalues[99] & 0xff;
operatorname[30]=0;
if (operatorname[0] != 0) {
```



```

        setcolor(BROWN);
        showline("msg_CustomL  ");
        outtextxy(40,380,msg_line);
        outtextxy(40,400,operatorname);
    }
}

void tellareas(char subarea,WINDOW *wnd_head)
{
    char name[25];
    if (subarea == 1) strcpy(name,"  Areas & Subareas");
    display_window(wnd_head);
    wprintf(wnd_head,"\n%s",name);
}

void tellmessage(char words[30], int count)
{
    WINDOW *wnd_message;
    wnd_message = establish_window(14,1,3,count);
    set_colors(wnd_message,ALL,RED,WHITE,DIM);
    display_window(wnd_message);
    wprintf(wnd_message,words);
    twobells();
    delay(2500);
    delete_window(wnd_message);
}

void noclientinfo()
{
    operationinfo("NoClien",10,5,nokey);
}

void readerror()
{
    operationinfo("ErrRdDD",10,5,nokey);
}

void notready()
{
    operationinfo("DDNORDY",10,5,nokey);
}

void tellfloppy()
{
    operationinfo("SelDD ",10,5,nokey);
}

```

```
}
```

```
void setpost()
{
    pointflag=1;
    pointcolor=EGAnum[postcolor];
}
```

```
void setbase()
{
    pointflag=0;
    outflag=0;
    outholdflag=0;
    pointcolor=EGAnum[basecolor];
}
```

```
/* Item Record */
```

```
void sortreport(int which)
{
    int i,j;
    unsigned int t,t1,t2;
    char temp[31];
    for (i=0; i<reportpoint+1; i++) {
        for (j=i+1; j<reportpoint+1; j++) {
            if (which == 'M') {
                t1=report[i].max;
                t2=report[j].max;
            }
            else {
                t1=report[i].fall;
                t2=report[j].fall;
            }
            if (t1 < t2) {
                strcpy(temp,report[i].name);
                strcpy(report[i].name,report[j].name);
                strcpy(report[j].name,temp);
                t=report[i].max;
                report[i].max = report[j].max;
                report[j].max=t;
                t=report[i].min;
                report[i].min = report[j].min;
                report[j].min=t;
                t=report[i].rise;
                report[i].rise = report[j].rise;
            }
        }
    }
}
```

```

        report[j].rise=t;
        t=report[i].fall;
        report[i].fall = report[j].fall;
        report[j].fall=t;
    }
}

}

void showreport()
{
    int j,rpoint,start,end,key,askkey;
    WINDOW *wnd_hold;
    if (reportflag) clear_window(wnd_report);
    setmenuhelp("list ",0,0);
    wnd_hold = establish_window(0,0,28,60);
    showline("msg_ItemReadings");
    set_title(wnd_hold,msg_line);
    set_colors(wnd_hold,ALL,letter2color,CYAN,DIM);
    set_colors(wnd_hold,ACCENT,accentcolor,WHITE,BRIGHT);
    display_window(wnd_hold);
    rpoint=0;
    start=0;
    key=0;
    while (key != ESC) {
        wcursor(wnd_hold,0,0);
        if (start+20 <= reportpoint) end = start+20;
        else end = reportpoint+1;
        for (j=start; j<end; j++)
        {
            if (j == rpoint) reverse_video(wnd_hold);
            wprintf(wnd_hold,"%#2d. %s",j+1,report[j].name);
            wprintf(wnd_hold," %#3d",report[j].max);
            wprintf(wnd_hold," %#3d",report[j].min);
            wprintf(wnd_hold," %#3d",report[j].rise);
            wprintf(wnd_hold," %#3d",report[j].fall);
            wprintf(wnd_hold," %s",report[j].potency);
            if (end != j) wprintf(wnd_hold,"\n");
            normal_video(wnd_hold);
        }
        wcursor(wnd_hold,10,22);
        wprintf(wnd_hold,"Items: %#3d Limit: %3d",reportpoint+1,nreport+1);
        key = get_char();
        key = toupper(key);
        switch(key) {

```

```

case HOME: rpoint=0;
           start=0;
           break;
case END: rpoint=reportpoint;
          start=rpoint-19;
          if (start < 0 ) start=0;
          break;
case RSWITCH:
case DN: rpoint++;
        if (rpoint > reportpoint) {
            rpoint=0;
            start=0;
        }
        if (rpoint > start+19) {
            start = rpoint;
            clear_window(wnd_hold);
        }
        break;
case LSWITCH:
case UP: rpoint--;
        if (rpoint < 0) {
            rpoint=reportpoint;
            start=reportpoint-19;
            if (start < 0) start=0;
        }
        if (rpoint < start) {
            start=rpoint-19;
            clear_window(wnd_hold);
            if (start < 0) {
                start=0;
                rpoint=0;
            }
        }
        break;
case PGUP: start=start-20;
          if (start < 0) start=0;
          clear_window(wnd_hold);
          rpoint=start+19;
          break;
case PGDN: start = start+20;
          if (start > reportpoint) start=0;
          clear_window(wnd_hold);
          rpoint=start;
          break;
case 'W': display_time(); break;

```

```

case 'M':
case 'F': sortreport(key);
           clear_window(wnd_hold);
           start=0;
           rpoint=0;
           break;
case 'P': pnum=0;
           printing(); break;
case DEL: if (reportpoint < 0) break;
           for (j=rpoint; j<reportpoint+1; j++) {
               strcpy(report[j].name,report[j+1].name);
               strcpy(report[j].potency,report[j+1].potency);
               report[j].max = report[j+1].max;
               report[j].min = report[j+1].min;
               report[j].rise = report[j+1].rise;
               report[j].fall = report[j+1].fall;
           }
           reportpoint--;
           if (rpoint > reportpoint) {
               rpoint--;
               start = rpoint-19;
               if (start < 0) start=0;
           }
           writeirecfile();
           beep();
           clear_window(wnd_hold);
           break;
case '-': askkey=operationinfo("delrcrd",5,5,yeskey);
           if (askkey == 'Y' ) {
               rpoint--;
               reportpoint = rpoint;
               writeirecfile();
               clear_window(wnd_hold);
           }
           break;
default: break;
}

}

delete_window(wnd_hold);
}

```

```

void reportone()
{
if (outid[0] == ' ') return;
display_window(wnd_report);
}

```

```

if (reportpoint == nreport) {
    showline("msg_FULL    ");
    wprintf(wnd_report,msg_line);
    twobells();
    return;
}
else {
    if (!max) return;
    ++reportpoint;
    strcpy(report[reportpoint].name,outname);
    strcpy(report[reportpoint].id,outid);
    strcpy(report[reportpoint].potency,pot[dilindex].name);
    report[reportpoint].max = max;
    report[reportpoint].min = min;
    report[reportpoint].rise = rise;
    report[reportpoint].fall = fall;
    beep();
    writeirecfile();
    wprintf(wnd_report,"\n%d. %s%d",reportpoint+1,outname,max);
}
}

```

```

void toggle_report()
{
    if (reportflag) {
        reportflag=0;
        delete_window(wnd_report);
    }
    else {
        reportflag=1;
        wnd_report = establish_window(41,5,3,39);
        showline("msg_ItemReadings");
        set_title(wnd_report,msg_line);
        set_colors(wnd_report,ALL,letter3color,WHITE,DIM);
        display_window(wnd_report);
    }
}

```

/* Program PCB */

```

void labelwaveplot(void)
{
    float floatj;
    int i;
    char string[5];

```

```

setviewport(526,0,549,281,1);
clearviewport();
setviewport(0,0,639,479,1);
settextstyle(SMALL_FONT,HORIZ_DIR,USER_CHAR_SIZE);
floatj=5.0;
if (wavepoint) {
    for (i=0; i<11; i++) {
        sprintf(string,"%-+#1.1f",floatj);
        outtextxy(526,i*25.5,string);
        floatj=floatj-1.0;
    }
}
else {
    for (i=0; i<11; i++) {
        sprintf(string,"%#1.1f",floatj);
        outtextxy(526,i*25.5,string);
        floatj=floatj-0.5;
    }
}
}

```

```

void plotwave(void)
{
float floatj;
int i,x=5;
int yshift;
floatj=5.0-(voltage/51.0);
if (voltage) yshift=261.0-(26.1*floatj);
else yshift=131;
setfillstyle(SOLID_FILL,WAVEBKCOLOR);
bar(2,2,518,266);
setcolor(LINECOLOR);
setlinestyle(SOLID_LINE,0,THICK_WIDTH);
if (wavepoint) {
    for (i=0; i<255; i++) {
        line(x,yshift-plotamp[i],x+1,yshift-plotamp[i+1]);
        x++;
    }
    for (i=255; i>0; i--) {
        line(x,yshift-plotamp[i],x+1,yshift-plotamp[i-1]);
        x++;
    }
}
else line(x,261-plotamp[i],x+512,261-plotamp[i]);
setcolor(TEXTCOLOR);
}

```

```

setlinestyle(SOLID_LINE,0,NORM_WIDTH);
labelwaveplot();
}

```

```

void howsprinter(void)
{
printerstatus = biosprint(2,0,pnum);
prtmsg = 4; /*unknown*/
if (printerstatus == 144) prtmsg=0; /*ready*/
if (printerstatus == 56 || printerstatus == 40) prtmsg=1; /*paper*/
if (printerstatus == 24 || printerstatus == 8) prtmsg=2; /*line*/
if (printerstatus == 136 || printerstatus == 128 ||
    printerstatus == 184 || printerstatus == 200) prtmsg=3; /*off*/
}

```

```

void sysstatus(void)
{
int key=0;
int xasp, yasp; /* Used to read the aspect ratio*/
struct viewporttype viewinfo; /* Params for inquiry procedures*/
struct palettetype palette;
char *driver, *mode; /* Strings for driver and mode */
int x,y;
unsigned memsize;
unsigned long mem_avail;
setgraphmode(g_mode);
setbkcolor(LIGHTGRAY);
Bibox(2,300,432);
setviewport(40,10,600,400,1);
DrawBorder();
operationgraph("YSMENU",5,22,nokey);
getpalette(&palette); /* Read the palette */
MaxColors = getmaxcolor() + 1; /* Read maximum number of colors*/
MaxX = getmaxx();
MaxY = getmaxy(); /* Read size of screen */
getviewsettings( &viewinfo );
getpalette( &palette );
x = 207;
y = 52;
setcolor(BLUE);
gprintf(&x,&y, "%d",nitem);
gprintf(&x,&y, "%d",nitem-narea);
gprintf(&x,&y, "%lu",npoint);
gprintf(&x,&y, "%lu",usepoint);
y += 5*textheight("H");

```



```

memsize = biosmemory();
gprintf(&x,&y,"%dK", memsize);
mem_avail = farcoreleft();
gprintf(&x,&y,"%lu",mem_avail);
y += 5*textheight("H");
howsprinter();
if (prtmsg == 0) showline("msg_PrinterReady");
if (prtmsg == 1) showline("msg_Outofpaper ");
if (prtmsg == 2) showline("msg_Notonline ");
if (prtmsg == 3) showline("msg_PrinterOff ");
if (prtmsg > 3) showline("msg_UnknownCode ");
gprintf(&x,&y,msg_line);
gprintf(&x,&y,"%d",printerstatus);
mem_avail = coreleft();
driver = getdrivename();
mode = getmodename(GraphMode); /* get current setting*/
y += 5*textheight("H");
gprintf(&x,&y,"%-20s (%d)",driver,GraphDriver);
gprintf(&x,&y,"%-20s (%d)",mode,GraphMode);
gprintf(&x,&y,"( 0, 0, %d, %d )",
getmaxx(),getmaxy());
gprintf(&x,&y,"( %d, %d, %d, %d )",
viewinfo.left,viewinfo.top,viewinfo.right,viewinfo.bottom);
gprintf(&x,&y,"%s",viewinfo.clip ? "ON" : "OFF" );
gprintf(&x,&y,"%d",MaxColors);
gprintf(&x,&y,"%d",getcolor());
showline("msg_PressESCkey ");
BottomLine(msg_line);
while (key != ESC) key = get_char();
restorecrtmode();
}

```

```

void tellprinter(void)
{
howsprinter();
if (prtmsg == 0) operationinfo("PtReady",10,5,nokey);
if (prtmsg == 1) operationinfo("Outpapr",10,5,nokey);
if (prtmsg == 2) operationinfo("Notline",10,5,nokey);
if (prtmsg == 3) operationinfo("PrtOff ",10,5,nokey);
if (prtmsg > 3 ) operationinfo("Unknown",10,5,nokey);
}

```

```

void readdefaults(void)
{
int j=0;

```

```

FILE *fptr;
if ((fptr = fopen("DEFAULT.DAT", "rb")) == NULL) {
    operationinfo("errorfl", 10, 5, yeskey);
    exit(0);
}
while (fread(&defaultvalues[j], sizeof(defaultvalues[50]), 100, fptr) == 1) j++;
fclose(fptr);
if (defaultvalues[2] == 0) defaultvalues[2] = 5;
if (defaultvalues[2] > 100) defaultvalues[2] = 5;
}

```

```

void getdefaults(void)
{
    int fhandle;
    struct stat info;
    savamp=defaultvalues[0];
    nscan=defaultvalues[1];
    timeout=defaultvalues[2];
    rate=defaultvalues[3];
    readinglow=defaultvalues[4];
    readinghi=defaultvalues[5];
    pageline=defaultvalues[6]-1;
    SerPort=defaultvalues[7];
    letter1color=defaultvalues[8];
    letter2color=defaultvalues[9];
    hicolor=defaultvalues[10];
    normcolor=defaultvalues[11];
    lowcolor=defaultvalues[12];
    bkcolor=defaultvalues[13];
    accentcolor=defaultvalues[14];
    basecolor=defaultvalues[15];
    postcolor=defaultvalues[16];
    areacolor=defaultvalues[17];
    itemcolor=defaultvalues[18];
    whichallergy=defaultvalues[19];
    clientdrive=defaultvalues[20];
    userptcode=defaultvalues[21];
    footflag=defaultvalues[22];
    language=defaultvalues[23] & 0xff;
    totalvisit=defaultvalues[25];
    totaltime=defaultvalues[26]*1000+defaultvalues[27];
    usedeletes=defaultvalues[28];
    useprints=defaultvalues[29];
    useA=defaultvalues[30];
    useB=defaultvalues[31];
}

```

```
vtype1 =defaultvalues[32];
vtype2 =defaultvalues[33];
vtype3 =defaultvalues[34];
vtype4 =defaultvalues[35];
amount =defaultvalues[36];
nPortDelay =defaultvalues[37];
nFootDelay =defaultvalues[38];
govern =defaultvalues[39];
restoreflag=defaultvalues[40];
multi_am =defaultvalues[41];
multi_vib =defaultvalues[42];
multi_egtyp=defaultvalues[43];
multi_ksr =defaultvalues[44];
multi =defaultvalues[45];
tone_ksl =defaultvalues[46];
tone_level =defaultvalues[47];
tone_dc =defaultvalues[48];
tone_dm =defaultvalues[49];
tone_fb =defaultvalues[50];
attack =defaultvalues[51];
decay =defaultvalues[52];
suslevel =defaultvalues[53];
retrate =defaultvalues[54];
rhy_mel =defaultvalues[55];
drum_BD =defaultvalues[56];
drum_SD =defaultvalues[57];
drum_TOM =defaultvalues[58];
drum_TCY =defaultvalues[59];
drum_HH =defaultvalues[60];
pitch =defaultvalues[61];
susON_OFF =defaultvalues[62];
keyON_OFF =defaultvalues[63];
octave =defaultvalues[64];
f_num8 =defaultvalues[65];
instrument =defaultvalues[66];
volume =defaultvalues[67];
percussion =defaultvalues[68];
bd_vol =defaultvalues[69];
hh_vol =defaultvalues[70];
sd_vol =defaultvalues[71];
tom_vol =defaultvalues[72];
tcy_vol =defaultvalues[73];
wavepoint =defaultvalues[74];
freqpoint =defaultvalues[75];
freqsave=freqpoint;
```

```

voltage =defaultvalues[78];
gain    =defaultvalues[79];
CARD     =defaultvalues[80];
PORTABLE =defaultvalues[81];
OUTPUTCOUNT = defaultvalues[83];
nGetBuffer = defaultvalues[84];
fhandle=open("C:\\WORK\\MASTER.DAT",O_WRONLY|O_BINARY);
fstat(fhandle,&info);
nitem=info.st_size/51;
close(fhandle);
fhandle=open("C:\\WORK\\FREQLIST.DAT",O_WRONLY|O_BINARY);
fstat(fhandle,&info);
freqcount=info.st_size/23;
close(fhandle);
}

```

```

void writedefaults(void)
{
FILE *fptr;
if ((fptr = fopen("DEFAULT.DAT","wb")) == NULL) {
    operationinfo("notfile",10,5,yeskey);
    exit(0);
}
else fwrite(defaultvalues,sizeof(defaultvalues[50]),100,fptr);
fclose(fptr);
}

```

```

void restoredefaults(void)
{
int i;
defaultvalues[0]=15; /*savamp*/
defaultvalues[1]=1024; /*nscan*/
defaultvalues[2]=6; /*timeout*/
defaultvalues[3]=3; /*rate*/
defaultvalues[4]=55; /*readinglow*/
defaultvalues[5]=45; /*readinghi*/
defaultvalues[6]=54; /*pageline*/
defaultvalues[7]=0; /*SerPort*/
defaultvalues[8]=RED; /*letter1*/
defaultvalues[9]=BLUE; /*letter2*/
defaultvalues[10]=RED; /*hi*/
defaultvalues[11]=GREEN; /*norm*/
defaultvalues[12]=YELLOW; /*low*/
defaultvalues[13]=LIGHTGRAY; /*back*/
defaultvalues[14]=BLACK; /*accent*/
}

```

```
defaultvalues[15]=RED; /*base*/
defaultvalues[16]=BLUE; /*post*/
defaultvalues[17]=RED; /*area*/
defaultvalues[18]=BLUE; /*item*/
defaultvalues[19]=0; /*whichallergy*/
defaultvalues[20]='B'; /*clientdrive*/
defaultvalues[21]='A'; /*userptcode*/
defaultvalues[22]='F'; /*footflag*/
defaultvalues[23]=language;
defaultvalues[24]=0; /*available xxx*/
defaultvalues[25]=0; /*totalvisit*/
defaultvalues[26]=0; /*totaltime*/
defaultvalues[27]=0; /*totaltime*/
defaultvalues[28]=0; /*usedeletes*/
defaultvalues[29]=0; /*useprints*/
defaultvalues[30]=0; /*useA*/
defaultvalues[31]=0; /*useB*/
defaultvalues[32]=100; /*vtype1*/
defaultvalues[33]=75; /*vtype2*/
defaultvalues[34]=50; /*vtype3*/
defaultvalues[35]=25; /*vtype4*/
defaultvalues[36]=0; /*amount*/
defaultvalues[37]=10; /*nPortDelay*/
defaultvalues[38]=150; /*nFootDelay*/
defaultvalues[39]=99; /*govern*/
defaultvalues[40]=' '; /*restoreflag*/
defaultvalues[41]=0; /*multi_am*/
defaultvalues[42]=0; /*multi_vib */
defaultvalues[43]=0; /*multi_egtyp*/
defaultvalues[44]=0; /*multi_ksr*/
defaultvalues[45]=0; /*multi*/
defaultvalues[46]=1; /*tone_ksl*/
defaultvalues[47]=0; /*tone_level*/
defaultvalues[48]=0; /*tone_dc*/
defaultvalues[49]=0; /*tone_dm*/
defaultvalues[50]=0; /*tone_fb*/
defaultvalues[51]=0; /*attack*/
defaultvalues[52]=0; /*decay*/
defaultvalues[53]=1; /*suslevel*/
defaultvalues[54]=1; /*relrate*/
defaultvalues[55]=0; /*rhy_mel*/
defaultvalues[56]=0; /*drum_BD*/
defaultvalues[57]=0; /*drum_SD*/
defaultvalues[58]=0; /*drum_TOM*/
defaultvalues[59]=0; /*drum_TCY*/
```

```

defaultvalues[60]=0; /*drum_HH*/
defaultvalues[61]=0; /*pitch*/
defaultvalues[62]=0x20; /*susON_OFF*/
defaultvalues[63]=0x10; /*keyON_OFF*/
defaultvalues[64]=4; /*octave*/
defaultvalues[65]=1; /*f_num8*/
defaultvalues[66]=7; /*instrument*/
defaultvalues[67]=0; /*volume*/
defaultvalues[68]=0; /*percussion*/
defaultvalues[69]=0; /*bd_vol*/
defaultvalues[70]=0; /*hh_vol*/
defaultvalues[71]=0; /*sd_vol*/
defaultvalues[72]=0; /*tom_vol*/
defaultvalues[73]=0; /*tcy_vol*/
defaultvalues[74]=0; /*wavepoint*/
defaultvalues[75]=0; /*freqpoint*/
defaultvalues[78]=255; /*voltage*/
defaultvalues[79]=0; /*gain*/
defaultvalues[80]=KEYCARD-1; /*CARD*/
defaultvalues[81]=KEYPORT; /*PORTABLE*/
defaultvalues[83]=200; /*count for skipping output */
defaultvalues[84]=3;
writedefaults();
}

```

```

void changecolors(void)
{
WINDOW *wnd_color;
int j,sel,c1;
char buffer[4];
setmenuhelp("colors ",10,10);
wnd_color = establish_window(10,0,25,50);
showline("msg_ChangeColors");
set_title(wnd_color,msg_line);
set_colors(wnd_color,ALL,LIGHTGRAY,BLACK,DIM);
set_colors(wnd_color,ACCENT,RED,BLACK,BRIGHT);
display_window(wnd_color);
showline("msg_MainLetters ");
wprompt(wnd_color,4,0,msg_line);
showline("msg_OtherLetters");
wprompt(wnd_color,4,2,msg_line);
showline("msg_HighRange ");
wprompt(wnd_color,4,4,msg_line);
showline("msg_OptimalRange");
wprompt(wnd_color,4,6,msg_line);
}

```

```

showline("msg_LowRange  ");
wprompt(wnd_color,4,8,msg_line);
showline("msg_Background ");
wprompt(wnd_color,4,10,msg_line);
showline("msg_Accent  ");
wprompt(wnd_color,4,12,msg_line);
showline("msg_BaseReadings");
wprompt(wnd_color,4,14,msg_line);
showline("msg_PostReadings");
wprompt(wnd_color,4,16,msg_line);
showline("msg_AreaList  ");
wprompt(wnd_color,4,18,msg_line);
showline("msg_ItemList  ");
wprompt(wnd_color,4,20,msg_line);
showline("msg_StandardColr");
wprompt(wnd_color,4,22,msg_line);
sel=8;
c1=0;
while (c1 != ESC) {
    for (j=8; j<19; j++) {
        if (sel == j) {
            if (defaultvalues[j] == EGA_LIGHTGRAY)

set_colors(wnd_color,ALL,BLACK,EGAnum[defaultvalues[j]],DIM);
            else
set_colors(wnd_color,ALL,EGA_LIGHTGRAY,EGAnum[defaultvalues[j]],DIM);
            reverse_video(wnd_color);
            strcpy(buffer,"<<");
        }
        else {

set_colors(wnd_color,ALL,EGA_LIGHTGRAY,EGAnum[defaultvalues[j]],DIM);
            normal_video(wnd_color);
            strcpy(buffer," ");
        }
        wcursor(wnd_color,23,(j-8)*2);
        wprintf(wnd_color," %s %s",Colors[defaultvalues[j]],buffer);
    }
    normal_video(wnd_color);
    cursoroff();
    c1=get_char();
    c1=toupper(c1);
    switch(c1) {
        case UP: sel--;
                    if (sel < 8) sel=18;

```

```

        break;
    case DN: sel++;
            if (sel > 18) sel=8;
            break;
    case FWD: defaultvalues[sel]++;
            if (sel > 12) j=8;
            else j=15;
            if (defaultvalues[sel] > j)
                defaultvalues[sel]=0;
            break;
    case BS: defaultvalues[sel]--;
            if (defaultvalues[sel] < 0)
                if (sel > 16) defaultvalues[sel] = 8;
                else defaultvalues[sel]=15;
            break;
    case 'S': defaultvalues[8]=RED; /*letter1*/
            defaultvalues[9]=BLUE; /*letter2*/
            defaultvalues[10]=RED; /*hi*/
            defaultvalues[11]=GREEN; /*norm*/
            defaultvalues[12]=14; /*low*/
            defaultvalues[13]=LIGHTGRAY; /*back*/
            defaultvalues[14]=BLACK; /*accent*/
            defaultvalues[15]=RED; /*base*/
            defaultvalues[16]=BLUE; /*post*/
            defaultvalues[17]=RED; /*area*/
            defaultvalues[18]=BLUE; /*item*/
            break;
    default: break;
}

}

delete_window(wnd_color);
}

void programpcb(void)
{
    char save_screen[25*80*2];
    int key,i;
    char string[40];
    float floatnum;
    void panelmenu();
    helpfunc=panelmenu;
    gettext(1,1,80,25,save_screen);
    setgraphmode(g_mode);
    ifgraphic=2;
    iftgraph=0;

```



```

setbkcolor(PANELBACK);
setusercharsize(1,1,1,1);
setcolor(TEXTCOLOR);
rectangle(0,0,520,268); /* Wave Plot */
rectangle(0,290,575,385); /* Panel Bars */
setfillstyle(SOLID_FILL,BLUE);
bar(2,292,573,383);
rectangle(4,319,454,329); /* Voltage Bar*/
rectangle(4,369,454,379); /* Frequency Bar*/
rectangle(250,395,550,460); /* Sound Box */
bar(252,397,548,458);
showline("msg_VoltageVolts");
sprintf(string,msg_line);
outtextxy(200,296,string);
showline("msg_FrequencyHz ");
sprintf(string,msg_line);
outtextxy(200,345,string);
labelbar();
Bibox(2,2,422);
settextstyle(TRIPLEX_FONT,HORIZ_DIR,3);
showline("msg_ControlPanel");
outtextxy(50,440,msg_line);
setcolor(TEXTCOLOR);
settextstyle(SMALL_FONT,VERT_DIR,USER_CHAR_SIZE);
showline("msg_Voltage ");
outtextxy(546,120,msg_line);
settextstyle(SMALL_FONT,HORIZ_DIR,USER_CHAR_SIZE);
floatnum=0.0;
for (i=0; i<6; i++) {
    sprintf(string,"%#1.1f",floatnum);
    outtextxy(90*i+2,304,string);
    line(90*i+4,315,90*i+4,318);
    floatnum=floatnum+1.0;
}
labelwaveplot();
showline("msg_Gain ");
outtextxy(282,400,msg_line);
showline("msg_Instrument ");
outtextxy(340,400,msg_line);
tellgain();
tellfrequency();
tellinstrument();
whichwave();
while (key != ESC) {
    INPUT();
}

```

```

foot=footpedal();
if (kbhit() || foot>0) {
    if (foot>0) key=foot;
    else {
        key = get_char();
        key = toupper(key);
    }
    switch(key) {
        case 'W': selectwave();
                    break;
        case F5: squarewave();
                    whichwave();
                    break;
        case F6: sinewave();
                    whichwave();
                    break;
        case F7: trianglewave();
                    whichwave();
                    break;
        case F8: dcline();
                    whichwave();
                    break;
        case RSWITCH:
        case '+': if (voltage != 255) {
                    voltage++;
                    whichwave();
                }
                break;
        case LSWITCH:
        case '-': if (voltage != 0) {
                    voltage--;
                    whichwave();
                }
                break;
        case '0': voltage = 0;
                    whichwave();
                    break;
        case 'M': voltage = 128;
                    whichwave();
                    break;
        case 'L': voltage = 255;
                    whichwave();
                    break;
        case '5': voltage = 255;
                    whichwave();
    }
}

```

```

        break;
    case '4': voltage = 204;
        whichwave();
        break;
    case '3': voltage = 153;
        whichwave();
        break;
    case '2': voltage = 102;
        whichwave();
        break;
    case '1': voltage = 51;
        whichwave();
        break;
    case 'D': voltage = 64;
        whichwave();
        break;
    case '.':
    case '>': gain++;
        if (gain > 7) gain = 0;
        tellgain();
        break;
    case ',':
    case '<': if (gain == 0) gain = 7;
        else gain--;
        tellgain();
        break;
    case 'I': instrument++;
        if (instrument > 15) instrument = 0;
        tellinstrument();
        break;
    case 'UP': increasefreq();
        break;
    case 'DN': decreasefreq();
        break;
    case '6': timer0++;
        tellfrequency();
        break;
    case '^': timer0--;
        tellfrequency();
        break;
    case '7': timer1++;
        tellfrequency();
        break;
    case '&': timer1--;
        tellfrequency();

```

/*

```

        break;
    case '8': timer2++;
        tellfrequency(timer0,timer1,timer2);
        break;
    case '*': timer2--;
        tellfrequency(timer0,timer1,timer2);
        break;*/
    case HOME: topfrequency();
        break;
    case INS: afrequency();
        break;
    case END: lowfrequency();
        break;
    default: break;
}

}

defaultvalues[66]=instrument;
defaultvalues[74]=wavepoint;
defaultvalues[75]=freqpoint;
defaultvalues[78]=voltage;
defaultvalues[79]=gain;
writedefaults();
backtext();
puttext(1,1,80,25,save_screen);
iftgraph=1;
ifgraphic=0;
}

void defaults(void)
{
    WINDOW *wnd_defaults;
    int j,sel;
    static int limithi[] = {31,1024,600,10,100,50,80,10};
    static int limitlow[] = {0,2,1,1,50,0,40,0};
    int c=0;
    wnd_defaults = establish_window(0,0,24,70);
    showline("msg_SysDefaults ");
    set_title(wnd_defaults,msg_line);
    set_colors(wnd_defaults,ALL,RED,WHITE,DIM);
    set_colors(wnd_defaults,ACCENT,BLACK,WHITE,BRIGHT);
    display_window(wnd_defaults);
    showline("msg_VolumeColon ");
    wprompt(wnd_defaults,5,1,msg_line);
    showline("msg_InitialScan ");

```

```

wprompt(wnd_defaults,5,3,msg_line);
showline("msg_ReadingTime ");
wprompt(wnd_defaults,5,5,msg_line);
showline("msg_GraphTime ");
wprompt(wnd_defaults,5,7,msg_line);
showline("msg_LowMaximum ");
wprompt(wnd_defaults,5,9,msg_line);
showline("msg_HighMinimum ");
wprompt(wnd_defaults,5,11,msg_line);
showline("msg_LinesPerPage");
wprompt(wnd_defaults,5,13,msg_line);
showline("msg_Language ");
wprompt(wnd_defaults,5,16,msg_line);
showline("msg_AllergyList ");
wprompt(wnd_defaults,5,17,msg_line);
showline("msg_ClientDrive ");
wprompt(wnd_defaults,5,18,msg_line);
showline("msg_PointsList ");
wprompt(wnd_defaults,5,19,msg_line);
showline("msg_FootKey ");
wprompt(wnd_defaults,5,20,msg_line);
showline("msg_CforColors ");
wprompt(wnd_defaults,5,22,msg_line);
sel=0;
readdefaults();
setmenuhelp("default",18,3);
while (c != ESC) {
    for (j=0; j<7; j++) { /* 8 for fall */
        if (sel == j) reverse_video(wnd_defaults);
        else normal_video(wnd_defaults);
        wcursor(wnd_defaults,27,j*2+1);
        if (j != 2) wprintf(wnd_defaults, "%#4d", defaultvalues[j]);
        else wprintf(wnd_defaults, "%#5.1f", defaultvalues[2]*.1);
    }
    normal_video(wnd_defaults);
    wcursor(wnd_defaults,25,16);
    wprintf(wnd_defaults, "%s", Language[language-65]);
    wcursor(wnd_defaults,23,17);
    if (whichallergy) showline("msg_RinkelSeries");
    else showline("msg_StandardLow ");
    wprintf(wnd_defaults,msg_line);
    wcursor(wnd_defaults,28,18);
    wprintf(wnd_defaults, "%c", clientdrive);
    wcursor(wnd_defaults,28,19);
    wprintf(wnd_defaults, "%c", userptcode);
}

```

```

wcursor(wnd_defaults,22,20);
if (footflag == 'F') showline("msg_RightFootPed");
else showline("msg_DownArrowKey");
wprintf(wnd_defaults,msg_line);
if (PORTABLE == KEYPORT)
{
    cursor(50,2);
    printf("Size(Ww):%#3d",nGetBuffer);
    cursor(50,3);
    printf("Port(Xx):%#5d",nPortDelay);
    cursor(50,4);
    printf("Foot(Yy):%#5d",nFootDelay);
}
cursoroff();
c=get_char();
switch(c) {
    case UP: sel--;
        if (sel < 0) sel=6;
        break;
    case DN: sel++;
        if (sel > 6) sel=0;
        break;
    case FWD: if (sel==1) defaultvalues[sel] = defaultvalues[sel]*2;
        else defaultvalues[sel]++;
        if (defaultvalues[sel] > limithi[sel])
            defaultvalues[sel]=limitlow[sel];
        break;
    case BS: if (sel==1) defaultvalues[sel] = defaultvalues[sel]/2;
        else defaultvalues[sel]--;
        if (defaultvalues[sel] < limitlow[sel])
            defaultvalues[sel]=limithi[sel];
        break;
    case 'c':
    case 'C': changecolors();
        setmenuhelp("default",10,10);
        break;
    case 'l':
    case 'L': language++;
        language=readpulln(language);
        defaultvalues[23]=language;
        fclose(linefp);
        load_linetxt(language);
        break;
    case 'a':
    case 'A': if (whichallergy) whichallergy=0;

```

```

        else whichallergy=1;
        defaultvalues[19]=whichallergy;
        break;
case 'd':
case 'D': if (clientdrive == 'A') clientdrive = 'B';
        else clientdrive = 'A';
        defaultvalues[20]=clientdrive;
        break;
case 'p':
case 'P': if (userptcode < 'A') userptcode = '@';
        userptcode++;
        if (userptcode > 'z') userptcode = 'A';
        defaultvalues[21]=userptcode;
        break;
case 'f':
case 'F': if (footflag == 'F') footflag = 'K';
        else footflag = 'F';
        defaultvalues[22]=footflag;
        break;
case F10: programpcb();
        helpfunc=help;
        break;
case 'w':
case 'W':
        if (c == 'W') nGetBuffer++;
        else nGetBuffer--;
        if (nGetBuffer > 11) nGetBuffer = 0;
        if (nGetBuffer < 0) nGetBuffer = 11;
        defaultvalues[84] = nGetBuffer;
        break;
case 'x':
case 'X': if (c=='X') nPortDelay++;
        else nPortDelay--;
        if (nPortDelay > 50) nPortDelay = 0;
        if (nPortDelay < 0) nPortDelay = 50;
        defaultvalues[37]=nPortDelay;
        break;
case 'y':
case 'Y': if (c == 'Y') nFootDelay++;
        else nFootDelay--;
        if (nFootDelay < 0) nFootDelay = 600;
        if (nFootDelay > 600) nFootDelay = 0;
        defaultvalues[38]=nFootDelay;
        break;
default:break;

```

```

    }
}
getdefaults();
writedefaults();
delete_window(wnd_defaults);
}

void setup(void)
{
char outfo[100];
int key;
union REGS inregs, outregs;
set_ctl_brk(0);
readdefaults();
getdefaults();
delay(1000);
if (kbhit() != 0)
{
key = get_char();
key = toupper(key);
switch(key)
{
case 'P':
if (PORTABLE == 0x5050)
{
PORTABLE = 0x4F50;
printf("\nClear Portable");
}
else
{
PORTABLE = 0x5050;
printf("\nSet Portable");
}
defaultvalues[81] = PORTABLE;
writedefaults();
break;
case 'C':
if (CARD == 0x4343)
{
CARD = 0x4243;
printf("\nClear Card");
}
else
{
CARD = 0x4343;

```



```

        printf("\nSet Card");
    }
    defaultvalues[80] = CARD;
    writedefaults();
    break;
case 'S':
    if (SerPort)
    {
        printf("\nSet COM1");
        SerPort=0;
        defaultvalues[7] = 0;
        writedefaults();
    }
    else
    {
        printf("\nSet COM2");
        SerPort=1;
        defaultvalues[7] = 1;
        writedefaults();
    }
    break;
}
}
if (CARD != 0x4343 && PORTABLE != 0x5050)
{
    printf("\n\nNo CARD or PORTABLE requested: %x %x",CARD,PORTABLE);
    delay(1000);
    exit(0);
}
if (CARD == 0x4343)
{
    setPCB();
    offvolume();
    printf("\nDCM Card Connected: %x",CARD);
}
if (PORTABLE == 0x5050)
{
    printf("\nLooking for DCM Receiver at COM%d\n\n",SerPort+1);
    SetupDCM();
}
detectgraph(&g_driver, &g_mode);
if (g_driver < 0) {
    operationinfo("Nograph",10,4,iskey);
    exit(1);
}

```

```

if (registerfarbgifont(triplex_font_far) < 0) exit(1);
if (registerfarbgifont(small_font_far) < 0) exit(1);
initgraph(&g_driver, &g_mode, "");
g_error = graphresult();
if (g_error < 0) {
    printf("initgraph error: %s.\n", grapherrormsg(g_error));
    exit(1);
}
gettextsettings(&oldtext);
setbkcolor(bkcolor);
setcolor(BROWN);
settextstyle(TRIPLEX_FONT,HORIZ_DIR,5);
outtextxy(22,122,"EQ");
outtextxy(72,112,"4");
rectangle(14,119,620,220);
settextjustify(LEFT_TEXT,TOP_TEXT);
settextstyle(TRIPLEX_FONT,HORIZ_DIR,3);
outtextxy(22,162,"Electronic Questionnaire Fourth Generation");
settextstyle(TRIPLEX_FONT,HORIZ_DIR,1);
strcpy(outfo,code);
outfo[38]=0;
outtextxy(20,190,outfo);
settextjustify(oldtext.horiz,oldtext.vert);
getoperator();
Bibox(2,300,432);
delay(2500);
cleardevice();
if (!govern) {
    setcolor(BROWN);
    showline("msg_invest1  ");
    outtextxy(10,170,msg_line);
    showline("msg_invest2  ");
    outtextxy(10,190,msg_line);
    settextjustify(oldtext.horiz,oldtext.vert);
    settextstyle(oldtext.font,oldtext.direction,textsize);
    delay(2000);
}
//inregs.h.ah = 0;    /* 'get stat' service */
//inregs.x.dx = COM;   /* port number */
//inregs.h.al = CONF; /* configuration */
//int86(0x14, &inregs, &outregs);
restorecrtmode();
textbackground(bkcolor);
readareaf();
readfilter1();

```

```

readfilter2();
readpointfile();
allholdtags();
amp=savamp;
setscale();
helping=0;
pointcolor=EGAnum[basecolor];
cursoroff();
if (restoreflag == 'X') restorevisit();
defaultvalues[40] = 'X';
checkroom();
writedefaults();
if (CARD == 0x4343)
{
    changetimer(TIMER0,timer0);
    changetimer(TIMER1,timer1);
    changetimer(TIMER2,timer2);
    getselectwave();
}
}

void addstats()
{
    int key;
    key=operationinfo("upvstat",10,5,iskey);
    if (key == 'Y') {
        totalvisit++;
        totaltime=totaltime+tused;
        defaultvalues[25]=totalvisit;
        defaultvalues[26]=totaltime/0x01000;
        defaultvalues[27]=totaltime - defaultvalues[26];
        defaultvalues[31]++;
    }
    start_time();
    wcursor(wnd_timing,16,3);
    tused=0;
    wprintf(wnd_timing,"%3.1f minutes  ",tused/60);
}

void clear_visitstats(int otherflag)
{
    int key;
    key=operationinfo("ClearVt",20,4,iskey);
    if (key == 'Y') {
        defaultvalues[25] = 0; /* totalvisits */
    }
}

```

```

        defaultvalues[26] = 0; /* totaltimehigh */
        defaultvalues[27] = 0; /* totaltimelow */
        defaultvalues[32] = 100; /* vtypeA */
        defaultvalues[33] = 75; /* vtypeB */
        defaultvalues[34] = 50; /* vtypeC */
        defaultvalues[35] = 25; /* vtypeD */
        defaultvalues[36] = 100; /* amount */
        if (otherflag) {
            defaultvalues[28] = 0; /* usedeletes */
            defaultvalues[29] = 0; /* useprints */
            defaultvalues[30] = 0; /* useA */
            defaultvalues[31] = 0; /* useB */
        }
        useA++;
    }
}

void timingwindow(int otherflag)
{
    getdefaults();
    showline("msg_Totaltime ");
    wprompt(wnd_timing, 16, 7, msg_line);
    wprintf(wnd_timing, "%4.1f", totaltime/60);
    showline("msg_minutes ");
    wprompt(wnd_timing, msg_line);
    showline("msg_Totalvisits ");
    wprompt(wnd_timing, 14, 8, msg_line);
    wprintf(wnd_timing, "%d", totalvisit);
    showline("msg_Averagetime ");
    wprompt(wnd_timing, 4, 9, msg_line);
    if (totalvisit == 0) wprintf(wnd_timing, "0.0 ");
    else wprintf(wnd_timing, "%4.1f", (totaltime/60)/totalvisit);
    showline("msg_minutes ");
    wprompt(wnd_timing, msg_line);
    if (otherflag) {
        wprompt(wnd_timing, 6, 10, "1: ");
        wprintf(wnd_timing, "%d", usedeletes);
        wprompt(wnd_timing, 16, 10, "2: ");
        wprintf(wnd_timing, "%d", useprints);
        wprompt(wnd_timing, 26, 10, "3: ");
        wprintf(wnd_timing, "%d", useA);
    }
}

void timeinfo()

```

```

{
    struct tm *tm_now;
    long secs_now;
    long past_secs=0;
    char *str_now;
    int oflag=0;
    int key=0;
    wnd_timing = establish_window(10,4,13,60);
    showline("msg_VisitTime ");
    set_title(wnd_timing,msg_line);
    set_colors(wnd_timing,ALL,6,BLACK,DIM);
    set_colors(wnd_timing,ACCENT,WHITE,BLACK,DIM);
    display_window(wnd_timing);
    amount=vtype1;
    showline("msg_NOWTIME ");
    wprompt(wnd_timing,6,1,msg_line);
    showline("msg_STARTTIME ");
    wprompt(wnd_timing,6,3,msg_line);
    showline("msg_DURATIONTIME");
    wprompt(wnd_timing,6,5,msg_line);
    setmenuhelp("timing ",10,10);
    timingwindow(oflag);
    while (key != ESC) {
        wcursor(wnd_timing,13,3);
        wprintf(wnd_timing," %s",ctime(&time_buffer.time));
        while (kbhit() == 0) {
            cursoroff();
            time(&secs_now);
            tm_now = localtime(&secs_now);
            str_now = asctime(tm_now);
            str_now[24] = 0;
            if (secs_now != past_secs) {
                wcursor(wnd_timing,10,1);
                wprintf(wnd_timing," %s ",str_now);
                past_secs=secs_now;
            }
            stop_time();
            tused = difftime(tstop,tstart);
            wcursor(wnd_timing,16,5);
            wprintf(wnd_timing," %3.1f",tused/60);
            showline("msg_minutes ");
            wprintf(wnd_timing,msg_line);
        }
        key = get_char();
        key = toupper(key);
    }
}

```

```
switch(key) {  
    case DEL: clear_visitstats(oflag); break;  
    case '|': oflag = 1; break;  
    case INS: addstats(); break;  
    default: break;  
}  
writedefaults();  
timingwindow(oflag);  
}  
delete_window(wnd_timing);  
}
```

```
void boogieout()  
{  
    SioPutc(SerPort, 0xBB);  
    stopwave();  
    defaultvalues[40] = '|';  
    writedefaults();  
    set_ctl_brk(1);  
    ClearDCM();  
    exit(0);  
}
```